UNIVERSITY OF TWENTE.

**Faculty of Electrical Engineering,
Mathematics & Computer Science**

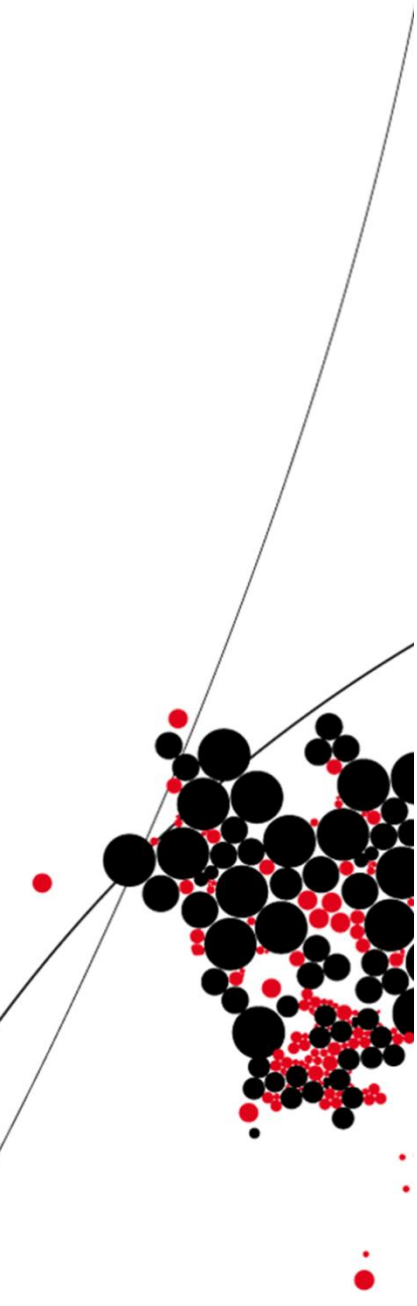
Realization of a Hardware Testbed for Noise-based, Frequency-offset Transmit Reference Systems

Dativa K. Tizikara
M.Sc. Thesis
August 2014

Supervisors:

Prof. dr. ir. ing. F.B.J Leferink
dr. ir. M.J. Bentum
dr. ir. A. Meijerink
I. Bilal, MSc.
dr. ir. R.A.R. van der Zee

Telecommunication Engineering Group
Faculty of Electrical Engineering,
Mathematics and Computer Science



Summary

Wireless sensor networks have become increasingly popular in the world today because of their applicability in monitoring and automation of home, environmental and industrial processes. As a result, they have generated a lot of research interest that focuses on techniques— such as transmit reference modulation— to achieve low cost-, low power- and robust operation in harsh radio environments. In transmit-reference modulation, the unmodulated carrier is transmitted together with the modulated signal, separated by a frequency shift or time delay.

The aim of this thesis was to develop a testbed using computer-based software radio peripherals, which would be used in current and future research as a flexible testing and platform to measure the physical layer performance of systems utilizing transmit-reference modulation.

A proof of concept simulation has been developed, and the bit error rate performance has been determined as a function of the signal to noise ratio per bit, for different spreading factors. Two cases have been investigated i.e. using a pseudo-noise sequence and additive white Gaussian noise as a carrier. In the first case, the simulation results have been shown to disagree with the theory due to the spectrum not being flat over the spreading bandwidth. In the second case, the simulation results have been shown to match the theory reasonably well.

Measurements carried out on the hardware testbed have shown the performance to be similar to the theoretical one at high spreading factors, but with a 2 dB shift. At lower spreading factors, it has been found to be better than predicted from theory as the signal to noise ratio per bit increases.

It has been found that local oscillator synchronization at the transmitter and receiver is required, unlike in conventional transmit-reference systems. Furthermore, the testbed has not achieved as much flexibility as expected in terms of the achievable bandwidths and spreading factors— partly due to hardware limitations, and partly due to the implementation that necessitated very high sampling rates and large data acquisitions.

These findings have led to the conclusion that the software radio peripherals provide a suitable testing platform for noise-based, frequency-offset, transmit reference systems. However, in order to exploit their full capability, it is recommended that the computer hardware be upgraded and some modifications be made in the software.

Contents

..... i

Summary i

Contents iii

1 Introduction 1

1.1 Background 1

1.2 Motivation 3

1.3 Goals of the Assignment 4

1.4 Related Work 4

1.5 Report Organization 4

2 Theory 5

2.1 Introduction 5

2.2 Baseband Representation of Bandpass Signals 5

2.3 The Universal Software Radio Peripheral (USRP) 7

2.3.1 Description 7

2.3.2 Operation 9

2.4 Spread Spectrum Communication 10

2.4.1 Direct Sequence Spread Spectrum (DSSS) 10

2.4.2 Advantages of spread spectrum 11

2.4.3 Disadvantages of Spread Spectrum Systems 12

2.5 Transmit Reference Modulation 14

2.5.1 Frequency Offset transmit reference method 14

2.5.2 Advantages of Transmit Reference modulation 16

2.5.3 Performance in Additive White Gaussian Noise 17

3 Simulations 21

3.1 Introduction 21

3.2 LabVIEW 21

3.3 Description of the Simulation 22

3.3.1 The Transmitter 23

3.3.2 Channel 25

3.3.3 The Receiver 26

3.3.4 Performance Analysis 27

3.4 Simulation Settings and Results 27
3.4.1 Waveforms and spectra 27
3.4.2 BER Performance 31
4 The Testbed 35
4.1 Introduction 35
4.2 Transmit Reference Modulation in the Passband 35
4.3 Realization of the Testbed 36
4.4 Configuration Steps for the USRP 37
4.4.1 The USRP in transmit mode 37
4.4.2 USRP in Receive Mode 38
4.5 Hardware Setup 39
4.6 Synchronization in the System 40
4.6.1 Local Oscillator Synchronization 40
4.6.2 Signal Alignment 41
4.7 Measurement Settings and Methodology 43
4.8 Challenges/Limitations of the Testbed 43
4.8.1 Achievable IQ Rates 43
4.8.2 Maximum Achievable Bandwidth 43
4.8.3 Achievable Spreading Factors 44
4.8.4 Manual measurement process 45
5 Results and Discussion 47
5.1 Measurement Results and Settings 47
5.1.1 Initial Measurement Results 47
5.1.2 Final Measurement Settings 49
5.1.3 Final Measurement Results 49
5.2 Discussion 50
6 Conclusions and Recommendations 53
6.1 Conclusions 53
6.2 Recommendations and Future work 54
References 57
Appendix A Block Diagram of Simulation 59
Appendix B Schematic Block Diagram of Testbed 61
B.1 Transmitter 61
B.2 Receiver 62

B.2.1 Receiver Signal Acquisition 62
B.2.2 Receiver Signal Processing 63
Appendix C User Manual 65
C.1 Simulation VIs 65
C.1.1 Final_simulation_cplxbaseband.vi 65
C.1.2 Final_simulation_cplxbaseband_formeasurements 67
C.2 Testbed VIs 68
Acknowledgements 71

Chapter 1

Introduction

This report details the work that has been carried out as part of a master assignment at the Telecommunication Engineering group of the University of Twente. In this chapter, the background and motivation for the research will be given and the specific research goals stated. This is followed by a brief discussion on previous research done on the subject, and finally, the structure of the rest of the report will be presented.

1.1 Background

Wireless sensor networks (WSNs) have become an integral part of the world today. A WSN consists of a number of nodes spread over a geographical area, which collect real world data and pass it to other nodes within the network, or to a central control location for analysis or further processing. They are used in a variety of applications including, but not limited to:

- monitoring of environmental conditions such as temperature, humidity, sound, vibrations and pressure. The collected data can be used to detect disasters such as forest fires, earthquakes and floods;
- structural and equipment monitoring which is useful in detecting faults and carrying out maintenance in a timely manner;
- healthcare to collect patient information such as heart rate, body temperature etc. which can facilitate quick diagnoses;
- industrial and home automation of processes such as energy control, security etc.

Usually the nodes operate in remote locations or in extreme environments, which makes maintenance difficult. Furthermore, they are usually small in size and cannot accommodate large batteries. As a result, low power consumption becomes a vital requirement for WSNs. Some of the methods to reduce power consumption are utilization of low power modulation techniques, reducing the amount of data to be transmitted e.g. through data compression, reducing the frequency of transmissions, and utilization of strict power control mechanisms such as sleep and power off [1].

Another crucial challenge for the nodes in WSNs is that they are required to maintain reliable communication in very harsh radio environments, for example in the presence of fading and interference in the wireless channel. This can be accomplished by using robust communication techniques such as spread spectrum, in which the data signal is spread over a large bandwidth by either multiplying it with a spreading sequence, or by frequency hopping. This enables operation at low power spectral densities (PSDs), making the signal appear noise-like to other systems sharing the same frequency band. The system is also made robust to narrowband interference (frequency selective fading, jamming) since only a small portion of the occupied spectrum is affected. The signal arriving at the receiver— which consists of the spread signal, noise and other interfering signals— is again multiplied by the spreading sequence to recover the information signal. This operation suppresses the narrow band interferers because they are in effect spread at the receiver, thus resulting in a good signal to interference ratio (SIR), which is expressed as a processing gain.

However, in order to despread the received signal, it has to be multiplied with an exact replica of the spreading sequence, which makes synchronization a crucial task. The synchronization process involves two steps; code acquisition, in which the two spreading codes i.e. the received and stored code are aligned in phase to within less than a chip duration [2], and code tracking to maintain the alignment. Code acquisition is an especially difficult task because it involves a search through all the possible offsets of the code sequence. The offset can either be in time — due to transmission and propagation delays or frequency — due to mismatch between the transmitter and receiver oscillators, or both. Moreover, the acquisition is done in low signal to noise ratio (SNR) conditions, in the presence of interferers and in the presence of channel fading [3]. It therefore becomes a complex, power- and time-consuming process, especially when large spreading factors are used. Long synchronization times are not desirable for low duty cycle transmissions because typically once a node is woken up, it has to set up the link, transmit its data and then it is switched off again. This makes it impractical to spend a significant portion of the time the node is powered on for synchronization rather than actual data transmission, not to mention the resulting receiver complexity and cost, all of which are not desirable for a WSN.

Transmit-reference modulation is a well-known technique that was introduced in the context of ultra-wideband, to counter the inherent synchronization problem and receiver complexity. In this method, the unmodulated carrier is transmitted together with the modulated signal, separated in

time or frequency, such that at the receiver, there is no need to regenerate a local copy of the carrier signal. The received signal is simply correlated with a frequency-shifted or delayed version of itself to obtain the information signal, resulting in a very simple receiver. It also provides several other benefits such as eliminating the need for channel estimation since both the reference and modulated signal encounter the same distortion in the channel. In addition, the fact that the receiver only needs to know the delay (or frequency shift) and symbol timing, implies that it is possible to replace the commonly used pulse-based carriers with any other bearer format including pure noise. Using noise as a carrier can further simplify the system since noise is easy to generate and possesses a flat spectrum. Hence there is no need for additional spectrum shaping techniques. Finally, it will eliminate the use of sinusoidal carriers which generally require very careful considerations in the form of stable oscillators and filters [4].

1.2 Motivation

The Short Range Radio (SRR) chair of the Telecommunication Engineering (TE) group at the University of Twente is carrying out research— under the WALNUT (Wireless Ad-hoc Links using robust Noise-based Ultra-wideband Transmission) project— on one such noise-based, frequency-offset, transmit-reference system for wireless sensor networks operating in a crowded spectrum, at very low power [5]. It is under this framework that the master assignment was undertaken. The specific goals of WALNUT are to develop suitable physical and MAC layer techniques for wireless ad-hoc links, and integrate the analog circuitry on chip. The development of the chip is a lengthy process, and once available, it will have some limitations in terms of the achievable parameters such as bandwidth and frequency offsets. Therefore, development of a testbed was proposed for this master assignment, using computer-based universal software radio peripherals (USRPs) to test the impact of the transmission medium on the system performance, and to verify the accuracy of the assumptions/approximations that have been made while performing theoretical analysis, without having to wait for the chip to be completed. Furthermore, it would provide a flexible testing platform over a broader range of parameters than the current chip was expected to provide, and be useful in the future when further modifications are made to the system.

1.3 Goals of the Assignment

The specific research goals were:

- to investigate whether the USRP is a suitable platform for testing/measuring the transmit reference technique;
- to carry out simulations and implement a hardware test bed using the USRP(s) to investigate the physical layer performance of the TR system in noise, interference and/or fading environments.

1.4 Related Work

Research into noise-based transmit-reference systems at the TE group has been on-going for a few years, having been first proposed by basing on a similar scheme employed in the optical domain called Coherence Multiplexing [6]. Theoretical performance analysis of such a system in in the presence of additive white Gaussian noise (AWGN) has been done for both time- and frequency-offset systems by Bekkaoui [7] and Shang [8] respectively. Proof of concept microwave testbeds have also been built, and some measurements carried out by Taban [9] and Balkema [10]. Furthermore, the performance of noise-based frequency-offset transmit-reference in frequency selective channels and in the presence of in-band interference has also been investigated in [11] and [12], respectively.

1.5 Report Organization

The rest of the report is organized as follows: Chapter Two will discuss some relevant theory, specifically about baseband representation of bandpass signals, spread spectrum, transmit-reference modulation, and introduce the hardware which was used for the assignment. Chapter Three will give a brief introduction to the software used for the simulations, describe the operation of the simulations done, and present some simulation results. This will be followed by a description of the hardware setup, its operation and limitations, and how the measurements were carried out in Chapter Four. The obtained results from measurements on the hardware testbed will then be presented and compared to the expected results from the theory in Chapter Five. Finally, conclusions and any recommendations for future work will be given in Chapter Six.

Chapter 2

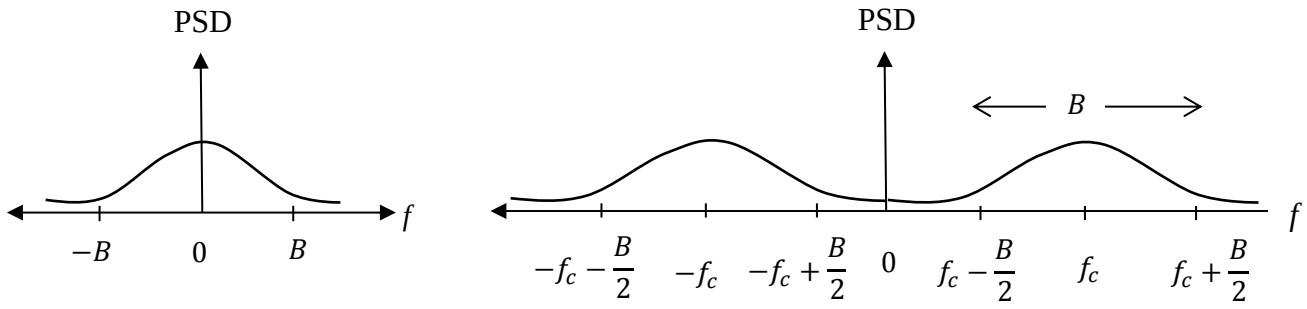
Theory

2.1 Introduction

This chapter will present the necessary theory on baseband signal representation, spread spectrum and transmit reference modulation, which is vital for understanding the work in this thesis. It will also give an introduction to the hardware that was used for the testbed.

2.2 Baseband Representation of Bandpass Signals

A baseband signal with bandwidth B is one whose one-sided power spectral density (PSD) is defined for a range of frequencies f between 0 and B, and approximately zero for all other frequencies. A bandpass signal on the other hand, is one whose PSD is centred around a certain frequency $f_c \neq 0$, and occupies a bandwidth B. The difference between these two signals is illustrated in Figure 2.1.



(a) (b)

Figure 2.1 Illustration of (a) baseband and (b) bandpass signals respectively

Wireless transmission of information is carried out over a certain range of frequencies in passband. This is achieved by modulation, in which an information signal (at baseband) is used to vary characteristics of a high frequency carrier. This results in a bandpass signal centred on the carrier frequency as in Figure 2.1(b). Modulation is necessary for several reasons such as;

- more bandwidth is available at higher frequencies.
- spectrum regulations dictate specific frequency bands within which different signals must be transmitted.
- Transmission medium properties may vary over a frequency range, hence making it more favorable to transmit the data in the corresponding band.

The translation to higher frequencies however brings about signal processing challenges, especially when it is a requirement to process the signals on a digital computer, because the highest frequency component of the signal $(f_c + B)$ is now much larger than the bandwidth. For example, accurate sampling of the signal to comply with the Nyquist sampling criterion means that the signal now has to be sampled at $2f_c + B$ compared to $2B$ in Figure 2.1. It therefore became necessary to form a baseband representation of band-pass signals.
In general, a bandpass signal $s(t)$ is represented by the equation

$$s(t) = s_I(t)\cos(2\pi f_c t) - s_Q(t)\sin(2\pi f_c t) \tag{2.1}$$

In this equation; f_c is the carrier frequency and $s_I(t)$ and $s_Q(t)$ are the in-phase and quadrature components of the modulated signal, respectively. These names follow from the fact that $s_I(t)$ is multiplied by the carrier (the cosine term) and $s_Q(t)$ is multiplied by a quadrature- phase version of the carrier [13] . A complex signal, called the complex envelope can then be defined as

$$\tilde{s}(t) = s_I(t) + js_Q(t) \tag{2.2}$$

The bandpass signal can be recovered from the complex envelope by multiplying with the complex exponential, $\exp(j2\pi f_c t)$ and taking the real part i.e.

$$s(t) = \mathbf{Re}[\tilde{s}(t)\exp(j2\pi f_c t)] \tag{2.3}$$

where **Re**[**.**] denotes real part of the expression in the square brackets.
From (2.3), it can be seen that the complex exponential only contains the carrier and all the message information in $s(t)$ is contained in the complex envelope. Hence no information is lost

by doing the processing on the complex envelope which is now at baseband. It should be noted however that the complex envelope is not a physical signal, but rather a mathematical one used to simplify the analysis. The scheme in Figure 2.2 shows how the quadrature components are formed from the bandpass signal in practice, and how the bandpass signal can be rebuilt from its quadrature components.

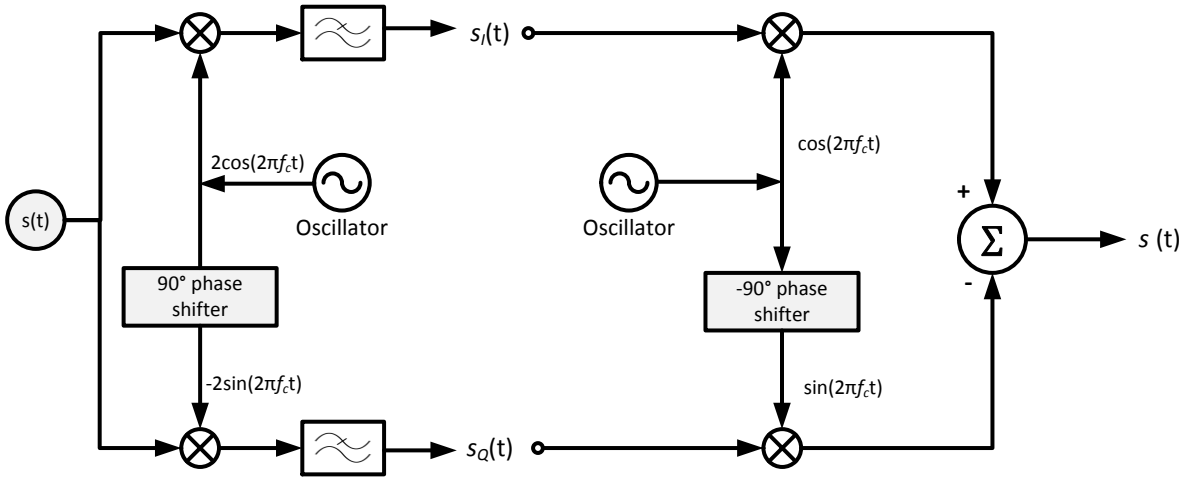


Figure 2.2 Scheme for generating the quadrature components of a bandpass signal and vice versa

2.3 The Universal Software Radio Peripheral (USRP)

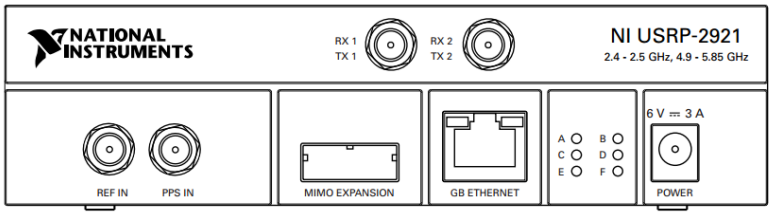
The USRP is a software-programmable, radio frequency (RF) transceiver developed by National Instruments (NI) as a platform for developing and prototyping wireless applications. It can be connected to a host computer, from where it can be configured and controlled with software such as NI LabVIEW[1]. In this way, it acts as a software defined radio with host-based digital signal processing capabilities [14].

2.3.1 Description

The model used for this assignment was the USRP 2921 and is shown in Figure 2.3. It is a half-duplex transceiver capable of operating in the 2.4 and 5.8 GHz ISM bands. It can provide up to 20 MHz (double-sided) baseband in-phase and quadrature (I/Q) bandwidth streaming at 25 mega samples per second (MS/s) for host-based processing with NI LabVIEW [14].

[1] An introduction to LabVIEW will be presented in Chapter 3.

(a)



(b)

Figure 2.3 (a) Image of the USRP 2921 and (b) detailed front panel **[15]**

The hardware front panel consists of:

- two ports (RX1/TX1 and RX2/TX2) fitted with SMA (Subminiature Version A) connectors, to which antennas or coaxial cables can be attached and used to either transmit or receive the RF signal;
- a gigabit (GB) ethernet port through which it is connected to a host computer(PC) and used to stream baseband I/Q samples to and from the PC;
- a MIMO expansion port which can be used to connect two USRPs together to achieve synchronization between devices in a master/slave configuration;
- a DC power input port;
- an input terminal for an external reference signal for the local oscillator (Ref in port). It accepts a 10 MHz sine or square wave with input power from 0-15 dBm;
- An input terminal for a pulse per second (PPS) timing reference (PPS in port).

A more detailed description of the USRP can be found in the USRP user manual [15].

2.3.2 Operation

The USRP essentially forms the digital baseband and intermediate frequency (IF) section of a typical communication system, (relying on the quadrature baseband concepts introduced in Section 2.1 to translate between the baseband and RF), as well handling the conversion between digital and analog. Its operation as a transceiver is shown in the block diagram in Figure 2.4.

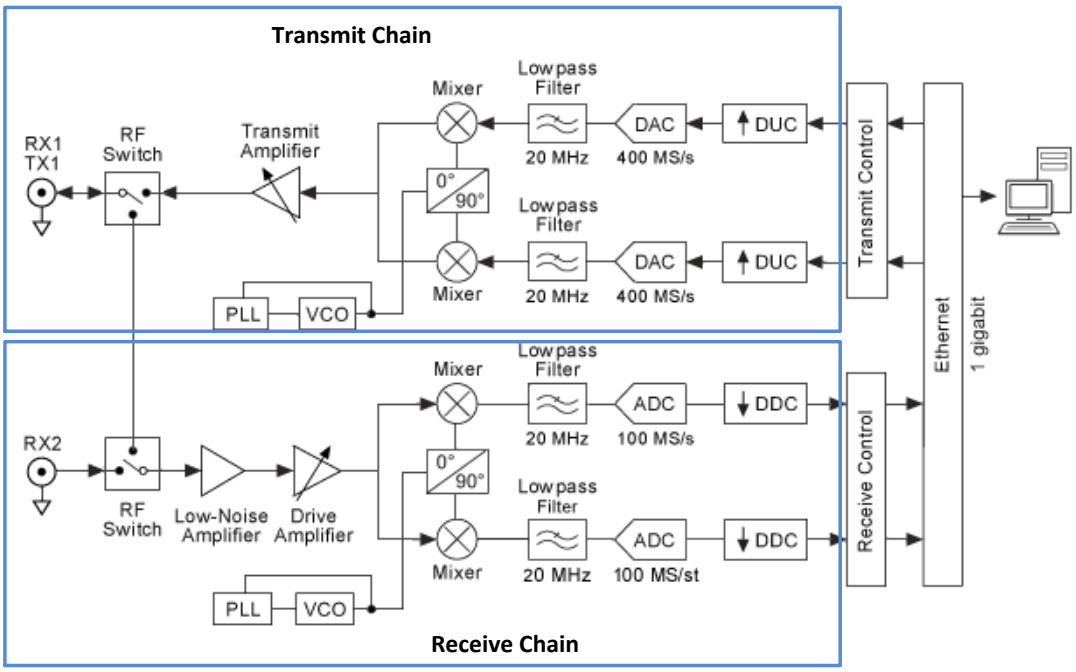


Figure 2.4 Detailed block diagram of the USRP illustrating its operation **[14]**

In the transmit chain, baseband in-phase and quadrature (IQ) samples are generated by the host computer and then transferred over the ethernet cable to the USRP at user specified rates up to 25 MS/s. The in-phase and quadrature samples are processed through parallel chains in which digital to analog conversion is done by 16-bit digital to analog converters (DACs). The resulting complex baseband IQ signals are filtered and then upconverted to the specified radio frequency. The signal can then be amplified by up to 30 dB, and transmitted over a channel. In the receiving mode, the process is reversed. The received signal is downconverted to baseband, filtered and digitized by 14-bit analog to digital converters (ADCs). The digitized I and Q samples are then transferred to the computer for further processing in the software e.g. LabVIEW.

2.4 Spread Spectrum Communication

Spread spectrum is a method of communication whereby a narrowband information signal is spread over a much larger bandwidth before transmission. The most commonly used methods are frequency hopping and direct sequence spread spectrum. In frequency hopping, the carrier frequency of the signal is continuously changed such that transmission is done in a different frequency band for a short period of time, whereas in direct sequence spread spectrum (DSSS), the information signal is multiplied by another signal that has a much larger bandwidth than the original signal. This section will focus on DSSS.

2.4.1 Direct Sequence Spread Spectrum (DSSS)

In the most general form, a low rate binary data sequence is multiplied by a higher rate pseudo random binary sequence, resulting in a signal whose total bandwidth is approximately equal to the bandwidth of the spreading signal [3]. The ratio of the bandwidth of the spreading signal to that of the data signal is known as the spreading factor. Since the power in the data signal is maintained constant, spreading it over a larger bandwidth means that the power spectral density (PSD) is lowered. If the spreading factor is large, the signal PSD can be below the noise power spectral density.

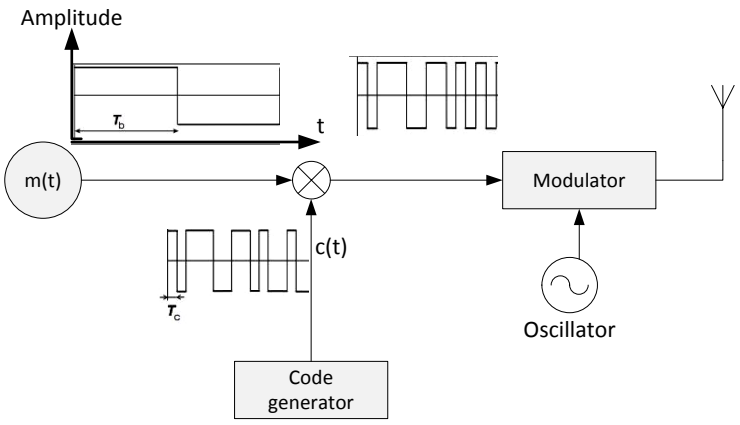


Figure 2.5 General form of spread spectrum transmitter

Figure 2.5 shows the basic spread spectrum transmitter. A low rate message signal $m(t)$ with bit duration T_b, is multiplied by a spreading code $c(t)$ with a much smaller duration T_c, called the chip duration. The resulting spreaded signal, which now has a bandwidth roughly equal to $1/T_c$,

can then be used to modulate a sinusoidal RF carrier. The corresponding spectrum is shown in Figure 2.6.

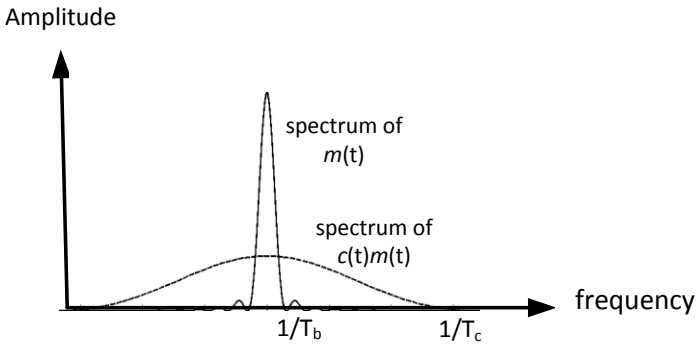


Figure 2.6 Spectra of narrowband message and spreaded signal

The received signal is despread by correlating it with the spreading sequence already known at the receiver. Spreading sequences are carefully chosen with excellent autocorrelation and cross-correlation properties such that a code is orthogonal to all other codes, as well as to time-shifted versions of itself. As a result, correlation of the received signal with any other code other than that used by the transmitter results in a value close to zero and no despreading occurs, whereas when the transmitter and receiver codes are synchronized, despreading can be achieved.

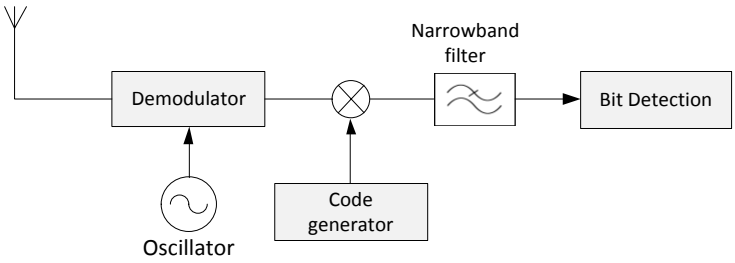


Figure 2.7 Basic DSSS receiver

2.4.2 Advantages of spread spectrum

Spread spectrum results in several benefits, some of which are:

I. **Immunity to interception**

In narrowband communication, the signal power is concentrated within a small range of frequencies such that the signal power is much higher than the noise floor. By searching within the frequency band of interest, an enemy can thus easily detect the signal. In

spread spectrum, the signal is spread over a very large bandwidth resulting in a very low power spectral density. Any intentional attack on the system such as jamming and eavesdropping becomes a difficult task.

II. **Spectrum sharing**

Spread spectrum facilitates spectrum sharing because of the low power spectral density of the transmissions. This enables other systems to share the same transmission bandwidth, without having to cope with interference form the spread spectrum signal because it appears as noise-like.

III. **Robust to narrowband interference**

Spectrum sharing means that the spread spectrum signal has to cope with interference from the existing narrowband signals. This is made possible by the fact that the narrowband interferers are spread at the receiver since they are multiplied by the spreading code during the despreading process. This lowers their power spectral density by a factor equal to the spreading factor hence improving the signal to interference ratio, and this is expressed as a processing gain. Then by applying the despread signal to a narrowband filter, most of the now-wideband interfering signal can be filtered out.

IV. Robustness to frequency selective fading

The signal being spread over a wide bandwidth means that frequency selective fading affects only a small portion of the spectrum.

2.4.3 Disadvantages of Spread Spectrum Systems

Like with any other technique, spread spectrum communications also has disadvantages.

I. Large Transmission Bandwidth

One obvious disadvantage is the large bandwidth required for transmission since the radio spectrum is a scarce resource. The channel also behaves differently over different frequencies. For example certain frequencies may be attenuated more than others. This varying nature of the channel response implies that accurate wideband channel modelling is required. In addition, the transceiver circuitry may also exhibit different characteristics at different frequencies.

II. **Increased system complexity**

Generally the system complexity and subsequently cost is increased with spread spectrum. For example accurate frequency synthesizers are required when frequency

hopping is used, and additional power control mechanisms need to be employed in case of multiple users.

III. **Synchronization**

Synchronization is perhaps the biggest disadvantage of spread spectrum because it is a crucial requirement for accurate recovery of the information signal. In essence, three levels of synchronization are needed i.e. code sequence synchronization, RF carrier synchronization and bit timing synchronization [3].

- **Code sequence synchronization**

This is required for the despreading operation. The pseudo-noise (PN) sequence is locally regenerated at the receiver and needs to be an exact replica of the version present in the received signal. It is accomplished in two steps:

- **PN sequence acquisition**

  In this step, some form of coarse alignment is achieved, typically to within a fraction of the chip duration [3]. The received signal is correlated with "all" possible delays of the locally generated PN code, then the peak values of the output are applied to a threshold detector, and the shift which produced the highest peak is taken as the code version which is most aligned with the received code. A decision as to whether synchronization has been attained can only be reached after all possible positions/shifts and the entire code period have been searched. For long codes (which are required for DSSS communications utilizing high spreading factors), it results in very long acquisition times and more complicated circuitry.

- **PN Tracking**

  After the PN sequence is acquired, the alignment needs to be maintained to obtain fine synchronization, and this is usually done with a delay locked loop.

- **RF carrier synchronization**

  Because the baseband spread spectrum signal is modulated onto an RF carrier for transmission, at the receiver, as in typical narrowband modulation techniques, it is required that the local oscillators at the receiver and transmitter are accurately synchronized in order to recover the baseband signal.

- **Correlation interval synchronization/ bit timing**

This is required for the bit detection process to indicate when the start and end of the integration should be, as well as the sampling time. This interval is usually equal to the data symbol period.

2.5 Transmit Reference Modulation

In conventional transceivers, sinusoidal carriers are used and modulated with the data – with the information being carried in the amplitude or phase of the carrier. Correct demodulation requires accurate carrier frequency and phase recovery at the receiver. Furthermore, in systems using spread spectrum, the spreading sequence also needs to be stored or replicated accurately in the receiver, which is often a difficult and time consuming task as discussed in Section 2.4.3. Transmit reference modulation is a technique that largely combats the synchronization problem by transmitting both the reference signal and the modulated signal over the same channel. The reference and modulated signal have to be separated in some way in order to be distinguishable, and this can be done by using a time delay or a frequency offset. The frequency offset method will be the subject of this thesis because separating the signals in frequency is a more attractive method than time delay since it is easier to obtain different frequency offsets by tuning the local oscillator, as compared to different delays which require changing the length of a delay element. In addition, a mixer-oscillator combination is easier to integrate on chip than long delay lines [16].

2.5.1 Frequency Offset transmit reference method

The frequency offset method relies on mixing either the modulated or reference signal with a low frequency sinusoidal signal from a low frequency oscillator. The basic (baseband) scheme of frequency offset transmit reference is shown in Figure 2.8.

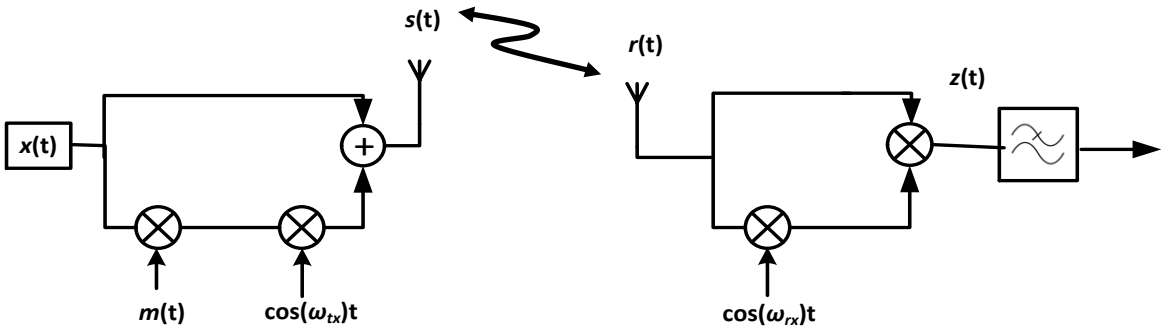


Figure 2.8 Baseband Scheme for transmit reference modulation

The bearer signal $x(t)$ is split into two: one branch carries the bearer as reference while in the second branch, the carrier is modulated with the message signal $m(t)$ and then a frequency offset applied. The frequency offset can be applied to either of the two branches. The two signals are combined and transmitted, resulting in a transmitted signal $s(t)$ given by

$$s(t) = x(t)[1 + m(t)\cos(\omega_{tx}t + \varphi_{tx})] \tag{2.4}$$

where $m(t)$ is the message signal, $x(t)$ the bearer signal, ω_{tx} is the angular frequency shift and φ_{tx} is the transmitter oscillator phase. At the receiver, the received signal is correlated with a shifted version of itself yielding

$$z(t) = r^2(t)\cos(\omega_{rx}t) \tag{2.5}$$

where the received signal $r(t)$ is given by $r(t) = s(t) + n(t)$. Ignoring the noise $n(t)$ for the time being and using (2.1)

$$z(t) = x^2(t)[1 + m(t)\cos(\omega_{tx}t + \varphi_{tx})]^2\cos(\omega_{rx}t + \varphi_{rx}) \tag{2.6}$$

$$= x^2(t)\Big[\cos(\omega_{rx}t + \varphi_{rx}) + m(t)\cos\big((\omega_{tx} + \omega_{rx})t + \varphi_{tx} + \varphi_{rx}\big) + m(t)\cos\big((\omega_{tx} - \omega_{rx})t + \varphi_{tx} - \varphi_{rx}\big) + \frac{1}{2}m^2(t)\cos(\omega_{rx}t + \varphi_{rx}) + \frac{1}{4}m^2(t)\cos\big((2\omega_{tx} + \omega_{rx})t + 2\varphi_{tx} + \varphi_{rx}\big) + \frac{1}{4}m^2(t)\cos\big((2\omega_{tx} - \omega_{rx})t + 2\varphi_{tx} - \varphi_{rx}\big)\Big] \quad (2.7)$$

From the third term of (2.7), it can be seen that if the frequency shift at the transmitter and receiver are equal i.e.$(\omega_{tx} = \omega_{rx})$, then the baseband message signal is recovered after filtering to suppress the terms at other frequencies. Furthermore, if the oscillator phases are synchronized, then the amplitude is maximum.

2.5.2 Advantages of Transmit Reference modulation

There are several advantages resulting from utilizing this scheme.

- First, there is no need to generate a local copy of the carrier at the receiver because it is transmitted with the modulated signal. Also, the fact that they undergo the same channel means that they experience the same distortion. In essence, no channel estimation or carrier recovery schemes are required at the receiver.
- When spread spectrum is used, $x(t)$ is the spreading sequence. From (2.7), it can be observed that the despreading occurs in the RF domain and the message directly appears at baseband. Furthermore, there is no need for the PN sequence to be known by the receiver. In order to despread the signal. Only the frequency offset needs to be known. This eliminates the need for code synchronization and essentially the synchronization procedure presented in Section 2.4.3 is reduced to only the bit timing. Furthermore, the fact that the receiver does not need to know the bearer signal implies that $x(t)$ can take on any pulse shape and can even be selected as a pure noise signal. Such a scheme using noise-based carriers is presented in [16], [4], [11]. Using pure noise as a carrier eliminates the need for spectrum shaping since it possesses a flat spectrum, and may also be easier to generate, while appearing as thermal noise to other systems [16].

- It results in a simple receiver structure since the complex channel estimation and carrier recovery circuitry is eliminated. Furthermore, from (2.6) it can be seen that the receiver simply requires squaring of the received signal and multiplication by the frequency offset.

However, a key point to note is that transmission of the reference in addition to the modulated signal results in penalty during transmission of at least 3 dB. However, for short range applications, this does not form an additional burden at the system level because more power is consumed in signal processing rather than in signal transmission [16].

2.5.3 Performance in Additive White Gaussian Noise

The performance of any real system is limited by several factors in the channel including noise, fading and interference. This section will summarize the performance of frequency offset transmit reference in the presence of white noise that has been analysed and presented in [8] and [16]. In order to control the power ratio between the modulated and reference signal, the reference can be multiplied by a constant c making the transmitted signal from (2.4)

$$s(t) = x(t)[c + m(t)\cos(\omega_{tx}t + \varphi_{tx})] \tag{2.8}$$

The message signal $m(t)$ is assumed to be a polar NRZ signal taking values ±1, and the bearer signal $x(t)$ is assumed to be a noise signal with a Gaussian distribution and zero mean. Then the PSD of $x(t)$ is flat, with a bandwidth B chosen to be much larger than the symbol rate of $m(t)$. In order to suppress the noise at the receiver, the received signal is first filtered. Assuming the channel and filter do not distort the transmitted signal, the received signal is then

$$r(t) = s(t) + n(t) \tag{2.9}$$

where $n(t)$ is the filtered noise.

Referring to Figure 2.8, the signal $z(t)$ after the mixer will consist of additional terms:

$$z(t) = s^2(t)\cos(\omega_{rx}t + \varphi_{rx}) + 2s(t)n(t)\cos(\omega_{rx}t + \varphi_{rx}) + n^2(t)\cos(\omega_{rx}t + \varphi_{rx}) \tag{2.10}$$

For simplification, $z(t)$ can be written as

$$z(t) = z_{ss}(t) + z_{sn}(t) + z_{nn}(t) \tag{2.11}$$

where $z_{ss}(t)$, $z_{sn}(t)$ and $z_{nn}(t)$ correspond to the first, second and third terms of (2.10), and will be called the signal-signal, signal-noise and noise-noise cross terms. $z_{ss}(t)$ corresponds to (2.6) and constitutes the desired message term, whereas $z_{sn}(t)$ and $z_{nn}(t)$ contribute to noise in the system. When the transmitter and receiver oscillators are synchronized, i.e. $\omega_{tx} = \omega_{rx}$ and $\varphi_{tx} = \varphi_{rx}$, and the oscillator frequencies are chosen to be either an integer multiple of or much larger than $2\pi/T_s$, then the signal to noise ratio (SNR) after integrate and dump filtering can be quantified by calculating the mean value of the output, which is shown to be proportional to the message signal [8]. Then the noise is calculated as the difference between the signal $z(t)$ and its expected value [8], [16]. The SNR was calculated in [8] and shown to be given by

$$SNR = \frac{16\,E_b/N_0}{\left(8c^2 + 36 + \frac{5}{c^2}\right)\frac{E_b/N_0}{S} + 2\left(8c^2 + 10 + \frac{3}{c^2}\right) + 2\left(4c^2 + 4 + \frac{1}{c^2}\right)\frac{S}{E_b/N_0}} \tag{2.11}$$

where in this equation: S is the spreading factor T_sB , and the ratio E_b/N_0 is the SNR per bit. From (2.11), there exists an optimum value of the factor c that maximizes SNR of the decision samples. It was shown [8] that choosing a value of $1/\sqrt{2}$ was an optimal choice for most practical situations. Furthermore, there also exists an optimal spreading factor for each value of E_b/N_0, which was shown to be $\approx 1.77\,E_b/N_0$. This results in a maximum SNR given by

$$SNR_{max} = \frac{2E_b/N_0}{5(\sqrt{2}+1)} \tag{2.12}$$

which is observed to be lower than that in a conventional coherent system which has $SNR = 2\,E_b/N_0$.

The disadvantage that comes with this is that it would be required to adaptively change the spreading factor whenever the E_b/N_0 changes, or require advance knowledge of the range of E_b/N_0 expected in the system so that a nearly optimal value can be chosen.

Chapter 3

Simulations

3.1 Introduction

This chapter gives an overview of the software and a description of the simulation, and presents the results obtained. Design of an engineering system can be summarized in three phases: theoretical analysis and modelling, simulation and hardware implementation. Simulation tries to predict the behaviour of a real world system or process, and provides a way to verify theoretical analysis before implementation on actual hardware. For the frequency offset transmit reference system, which is the subject of this thesis, simulations have previously been implemented in Simulink and Agilent's Advanced Design System, and presented in [8] and [10] respectively. The simulation environment used in this work was LabVIEW. The reason for this was that LabVIEW would be used to interact with the hardware in the testbed. Hence it was necessary to first simulate the system in the same environment as a proof of concept to verify that it would work, and that the individual building blocks had been implemented correctly.

3.2 LabVIEW

LabVIEW, short for Laboratory Virtual Instrumentation Engineering Workbench, is a graphical programming platform from National Instruments (NI) which is commonly used for interfacing with control and measurement devices, for example data acquisition devices, signal generators and many others used in engineering and laboratory applications. It provides support for a large number of instrument drivers — not limited to NI equipment, but also third party manufacturers — therefore making it a powerful tool when developing applications that depend on such hardware. LabVIEW provides a large number of signal processing, measurement, analysis and mathematical libraries consisting of individual blocks that perform specific functions, which can be used to build up a system. Furthermore, it avails additional toolkits that provide more advanced, application-specific functions such as digital filter design, RF and wireless communication, robotics, sound and vibration, etc.

A program in LabVIEW is called a Virtual Instrument (VI). To write such a program, a user interfaces with a virtual front panel on which data can be input — in the form of controls — and output — using indicators or graphs. The controls and indicators can be defined as numbers, characters or any other supported data type. The front panel is then connected to a schematic block diagram in which the 'code' is implemented as a series of blocks connected together by wires. The graphical programming approach reduces the complexity of text-based programming, making it easy to use for relatively inexperienced programmers. However, additional functionality is also provided to integrate text-based math and even to generate text-based code from a LabVIEW VI.

3.3 Description of the Simulation

The simulation developed is illustrated by the block diagram in Figure 3.1. It consisted of the three major components of any communication system:

- *the transmitter* in which a data signal is prepared for transmission.
- *the Channel* which was considered as an additive white Gaussian noise (AWGN) channel.
- *the receiver* in which the message is recovered from the received signal.

Once the system has been developed, performance measurement can then be done. In this case, the performance measure was a bit error rate calculation.

The detailed schematic block diagram, as implemented in LabVIEW is shown in Appendix A.

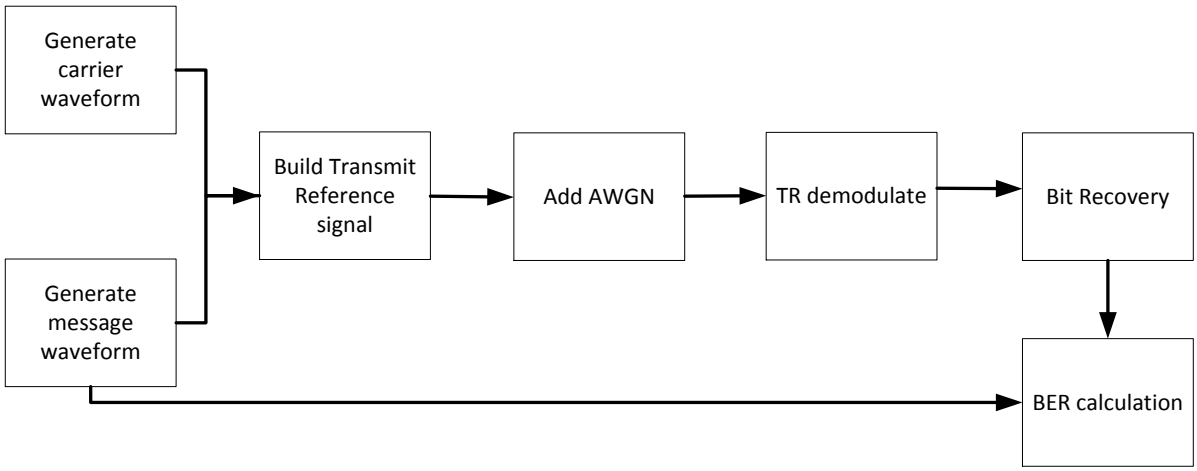


Figure 3.1 Block diagram of simulated transmit reference system

3.3.1 The Transmitter

In the transmitter, the message and carrier waveforms are generated and then used to build the transmitted reference signal.

- **Generate data and carrier**

  For this simulation, the message and carrier were both taken as polar non return to zero (NRZ) data signals and were built up in the following way.

  I. **Pseudorandom Noise (PN) sequence generator**

  This block generates a PN sequence of bits according to a Fibonacci, Galois or user-defined pattern. The selected bit-pattern is repeated until a specified number of bits are generated. The Fibonacci and Galois methods generate maximal length sequence (m-sequence) PN codes, using linear feedback shift registers[2]. The output of such a sequence is periodic with period $2^n - 1$, where n is the number of registers. The user can specify a PN order (number of shift registers) from $5-31$ and an initial state of the shift registers (an initial seed). In this simulation, the PN order was chosen as 31 in order to have long PN sequences. A different initial seed was selected for both the message and spreading sequence using a random number generator, and was changed every measurement to have more realistic results.

  II. **Unipolar to Polar conversion**

  The output of the PN sequence generator is a bit stream of 0s and 1s, which had to be mapped onto ±1 to form a polar NRZ sequence, using a for-loop and comparator.

  III. **Build waveform**

  All transmitted signals are analog, which implies that digital information has to be transformed into an analog format. Generally, this involves utilizing a specific pulse shape to build a waveform. A rectangular pulse was chosen because of its simplicity. It was built up with a simple array of 1s consisting of a user defined number of samples, spaced apart by the sampling period as shown in Figure 3.2. It can be observed that the initial sample is taken at the beginning of the bit period and the last sample at $(N-1)T_s$, where N is the number of samples in a chip period and T_s is the sampling period. This results in the final bit not being defined for an entire bit period. As a result, the last bit is lost after detection, because samples

[2] A detailed discussion on methods of PN sequence generation is out of the scope of this thesis.

are taken at NT_s. It was therefore necessary to append a few extra samples to the message before transmission. The easiest way to do this was to append some extra bits.

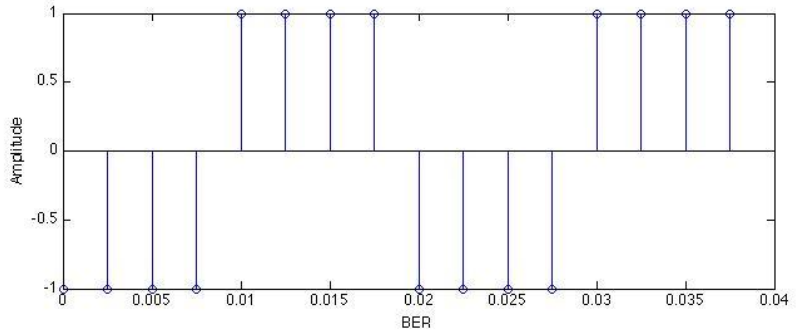


Figure 3.2 Illustration of sample signal waveform with T_b=0.01, T_s= 0.0025 and N = 4

The choice of the sampling period requires careful consideration i.e.

- it had to be chosen as $NT_s = T$, where T is the bit duration and N is an integer equal to the number of samples in the chip period;
- it had to comply with Nyquist theorem to prevent aliasing of the signal during processing. During TR demodulation the signal is squared which results in the bandwidth being doubled. Furthermore, after multiplying by the frequency offset ω_{tx}, cross terms are obtained at ω_{tx}, $2\omega_{tx}$ and $3\omega_{tx}$ from Equation (2.7). Hence if the spreading bandwidth is B, the sampling frequency should be chosen as at least $2\left(2B + \frac{3\omega_{tx}}{2\pi}\right)$

- **Build TR signal**

Once the data and carrier signal were formed, the transmit reference signal could then be built using a sinusoidal signal generator block and a direct implementation of Equation (2.4). In order to multiply two signals in LabVIEW, they are required to have the same sampling period, which was shown to depend on the chip duration in the previous section. It was therefore necessary to up-sample the message and cosine waveforms to the carrier sampling rate, which was done using a resampling block directly available in LabVIEW. The USRP, which would be used in the testbed, accepts as input the complex baseband samples of any signal applied to it as discussed in Section 2.3.2. It was therefore preferred to do a complex

baseband simulation in order to mirror the expected implementation of the hardware testbed as much as possible.

For passband operation, the transmit reference signal, (2.4) needs to be multiplied by a high frequency carrier ω_c to give

$$s(t) = x(t)[1 + m(t)\cos(\omega_{tx}t + \varphi_{tx})]\cos(\omega_c t) \quad (3.1)$$

Then using (2.1), the in-phase and quadrature components can be found as

$$s_I = x(t)[1 + m(t)\cos(\omega_{tx}t + \varphi_{tx})] \text{ and} \quad (3.2)$$

$$s_Q = 0$$

This shows that the complex baseband waveform corresponding to (3.1) consists of only a real part.

3.3.2 Channel

The channel model used was an additive white Gaussian noise channel and was implemented using an AWGN block from the modulation toolkit in LabVIEW. The block generates zero mean, complex AWGN, adds it to the input complex waveform and returns the signal plus noise as output. At the input, the user is required to define the:

- input complex signal, which is used to calculate the signal power (P_s)
- number of samples per symbol (N_s)
- number of bits per symbol (N_b)
- required E_b/N_0 in dB

The noise power (and subsequently variance) is then calculated using

$$P_{\text{noise}} = \frac{P_s}{\#bits} \times \left(\frac{E_b}{N_o}\right)^{-1}, \quad (3.3)$$

where *#bits* is the number of bits - given by $\frac{N_s}{N_b}$ and E_b/N_0 is in linear scale. The conversion from decibels to linear is also done within the block. The calculated variance is then used to

generate a Gaussian distributed pseudo random sequence using a modified version of the Box-Muller method[3]. The in-phase and quadrature components of the generated white noise need to be uncorrelated. This is accomplished by using two separate pseudo random generators with different initial seeds [17]. White noise by definition has infinite bandwidth. However, the generated noise is band limited to the bandwidth of the input waveform.

3.3.3 The Receiver

This section consisted of demodulating the transmit reference signal and bit detection.

- **TR demodulation**

  The transmitted signal, perturbed by AWGN, is obtained at the receiver, where it is demodulated by a direct implementation of (2.5). The received signal is squared and multiplied by the same frequency shift used in the transmitter using a squaring block, mixer and sinusoidal waveform generator, respectively.

- **Bit detection**

  From detection theory, it is well known that a matched filter is the optimal detector for maximizing the signal to noise ratio in the presence of Gaussian noise. Furthermore, when the correlation template is a rectangular pulse, an integrate and dump filter is an equivalent implementation of a matched filter.

  I. Integrate and dump implementation

  The integrate and dump function is not directly available in LabVIEW as a standard block, but was implemented with a discrete integrator and an additional algorithm to initialize the integrator to zero at every integer multiple of the symbol period. This required knowledge of the exact number of samples per symbol which could be easily calculated as

$$\text{samples per symbol} = \frac{N_c T_b}{T_c} = \frac{N_c \times \text{carrier bandwidth}}{\text{datarate}} = N_c S \qquad (3.4)$$

This equation follows from the fact that the message signal was up-sampled to the carrier sampling frequency as presented in Section 3.3.1. In (3.4), N_c is the number of samples in a chip period, T_c and T_b are the chip and bit period respectively and S is the spreading factor. The first equality follows from the fact that binary phase shift keying (BPSK) was assumed by

[3] The Box-Muller transform is one of several methods used to generate Gaussian random numbers.

considering a polar NRZ data signal. In theory, the integrator attains its maximum value at the end of a bit period.

II. Sample and hold

The next step in the bit detection process is to sample the integrate and dump output at multiples of the symbol period. The sample values are then applied to a comparator which returns a 1 or 0 depending on whether the sampled value is greater or less than zero respectively.

3.3.4 Performance Analysis

The performance measure in a digital system is the bit error rate, which expresses the percentage of error bits in a received data stream.

BER calculation

In this simulation, it is calculated by simply comparing the array of the input bits with that of the recovered bits and counting the erred bits. This gives an estimated BER given by

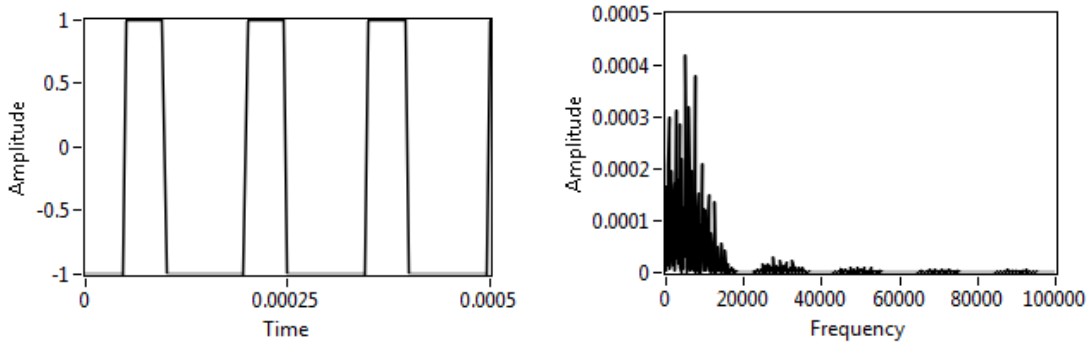
$$BER = \frac{\text{number of errored bits}}{\text{total number of input bits}} \tag{3.5}$$

3.4 Simulation Settings and Results

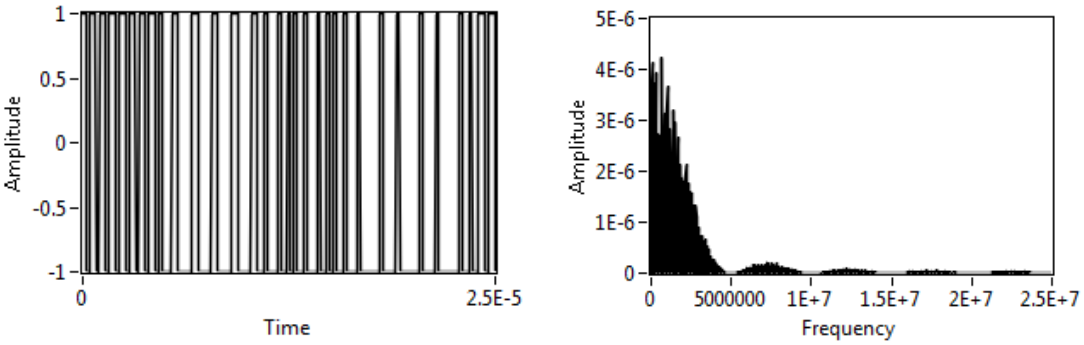
Some results of the developed simulation showing the output waveforms and spectra at different points in the system, as well as BER performance are presented.

3.4.1 Waveforms and spectra

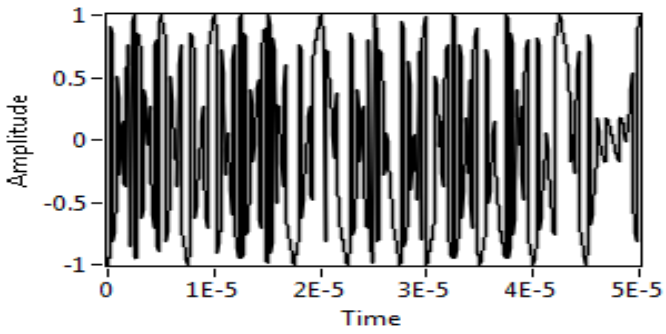
The illustration of signal waveforms and spectra was useful to verify that the simulation was working as expected at different points. The results below were obtained for 500 data bits, a datarate of 20 kbps, a carrier bandwidth of 5 MHz and a frequency shift of 200 kHz, and using 10 samples per symbol.



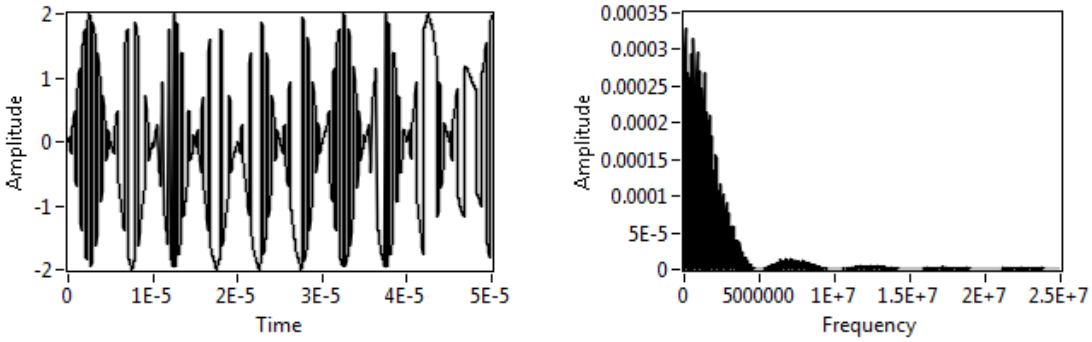
(a) Data signal (showing 10 bits) and its power spectrum



(b) Spreading sequence bits shown for half a symbol period and corresponding carrier spectrum

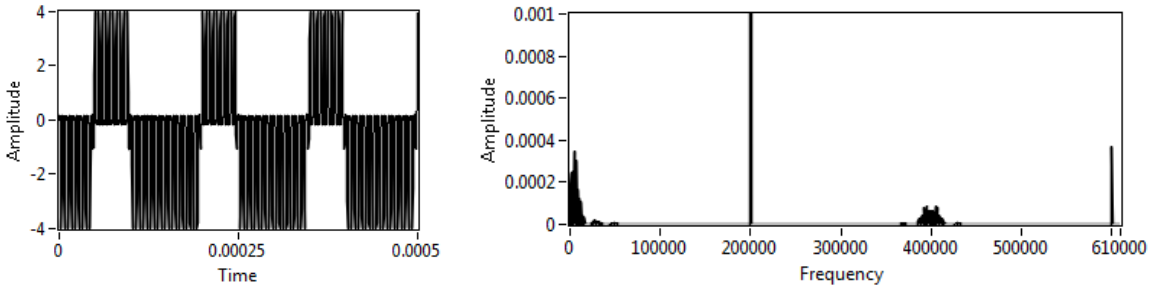


(c) Data × carrier × frequency offset for one symbol period

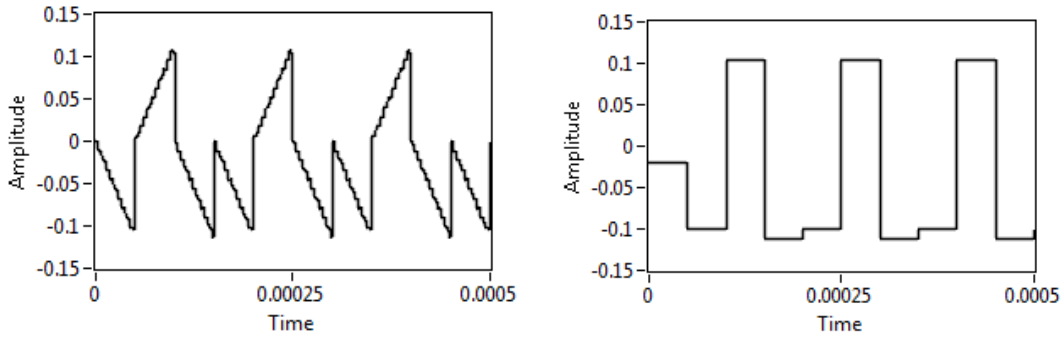


(d) Transmitted signal shown for one symbol period and corresponding spectrum

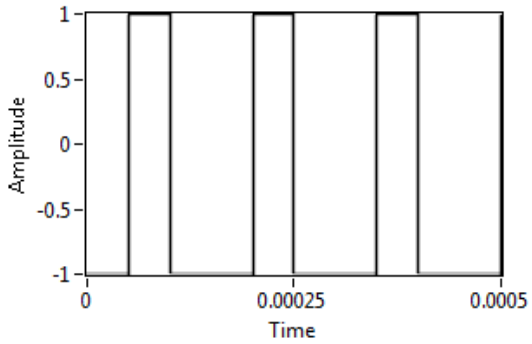
Figure 3.3 Illustration of signal waveforms and spectra at different points in the transmitter



(a) Signal after TR demodulation (for 10 symbol periods) and corresponding spectrum after TR demodulation



(b) Output of integrate and dump filter and sample and hold respectively



(c) Recovered bits

Figure 3.4 Illustration of signal waveforms and spectra at different points in the receiver

The waveforms and spectra shown are easily verified because they are simply numerical evaluations of the equations presented in the theory and their respective Fourier transforms. The spectra of the data and carrier waveforms extend over a very large bandwidth (ideally infinite from theory because they are sinc- squared functions), but for purposes of clear representation, only a small portion is shown. In Figure 3.5(a), it can be seen that the demodulated signal consists of the message at baseband and spectral components at multiples of the frequency shift as expected from Equation (2.7).

3.4.2 BER Performance

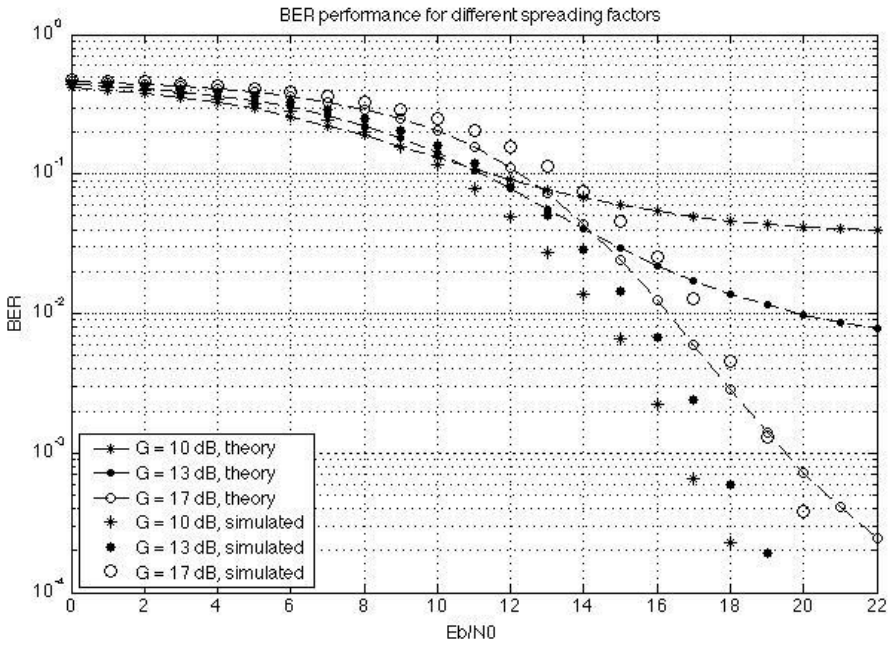
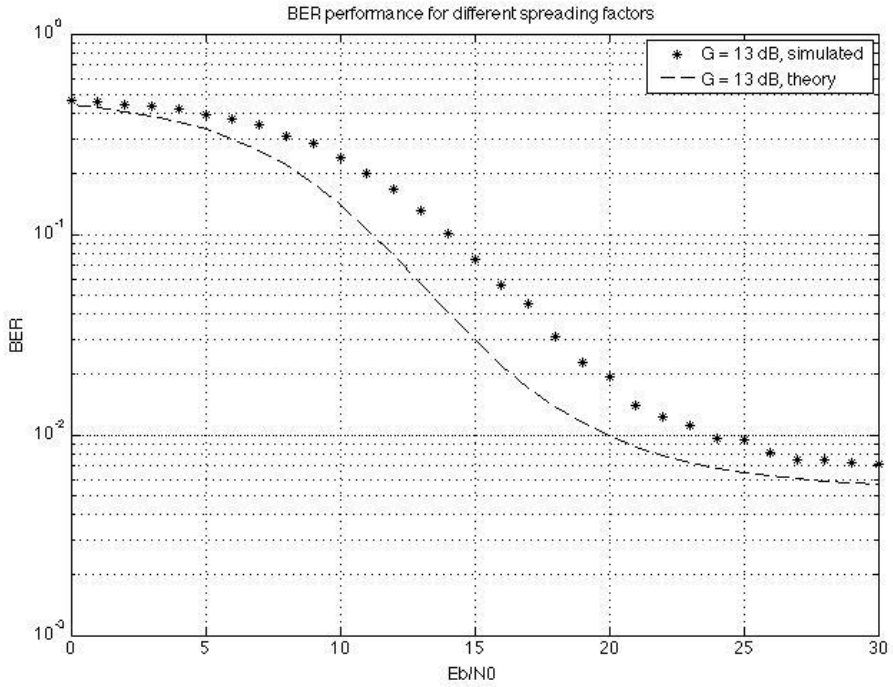

BER performance for different spreading factors
BER
Eb/N0
G = 10 dB, theory
G = 13 dB, theory
G = 17 dB, theory
G = 10 dB, simulated
G = 13 dB, simulated
G = 17 dB, simulated

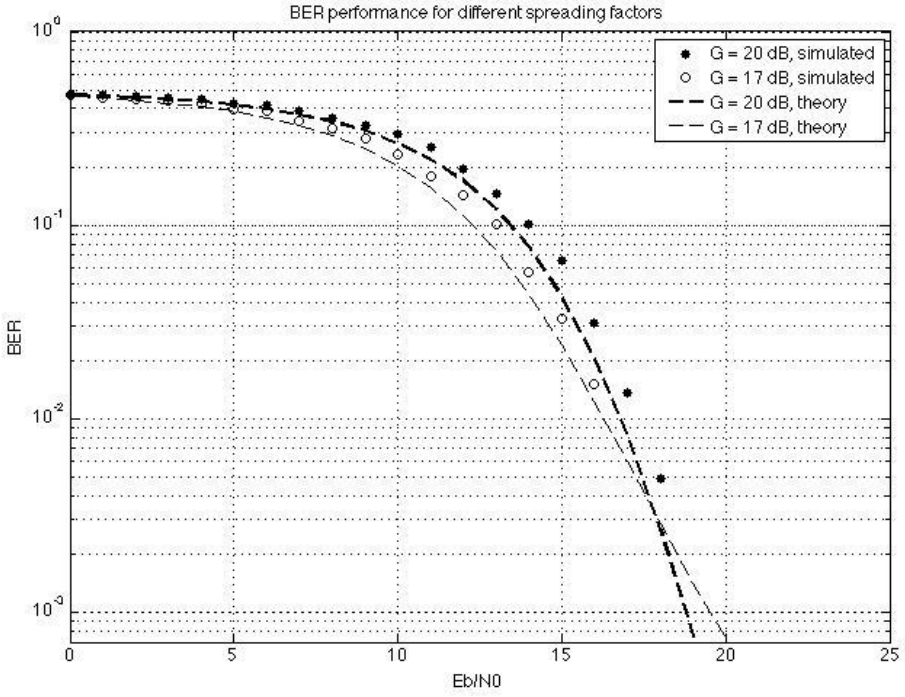
Figure 3.5 BER performance for different processing gains for polar NRZ spreading sequence

From Figure 3.5, it can be seen that the results do not agree with the theory. This is probably because in the theory, the reference spreading signal is assumed to have a flat power spectral density. When using a rectangular pulse, as in this case, the PSD is not flat.

It was therefore necessary to repeat the simulations with a noise carrier, which was generated with an AWGN block, and then filtered to limit the bandwidth to the required spreading bandwidth. The results in Figure 3.6 show the performance obtained for a 1 MHz bandwidth and different spreading factors—which were obtained by varying the datarate. From the theory, the frequency offset is required to be an integer multiple of the datarate or much larger. For these simulations, the frequency offset was chosen to be equal to the datarate to minimise the resulting signal bandwidth, and hence the required sampling rates as discussed in Section 3.3.1.



(a)



(b)

Figure 3.6 BER performance for different processing gains with noise as spreading sequence

In Figure 3.6, the dashed lines show the theoretical curves while the simulated values are shown with markers. When the spreading signal was changed to a white noise signal, the expected behaviour was observed i.e. at low values of E_b/N_0, lower spreading factors perform better and at higher values of E_b/N_0, the curves fall more steeply as the spreading factor increases. Furthermore, an error floor for the lower processing gain has been observed as expected from theory.

At higher spreading factors (Figure 3.6(b)), the obtained results match reasonably well with the theory but do not fit the curves exactly. This could be due to differences in the numerical computations in the software versus the analytical approximations. Similar simulations of the noise-based frequency offset transmit reference system done by Shang [8] and Balkema [10], also showed that the simulated values did not produce an exact fit. At lower spreading factors (Figure 3.6(a)), the simulated performance is worse than the theoretical performance.

In conclusion, the simulation results showed that the system could work and could then be implemented in the hardware.

Chapter 4

The Testbed

4.1 Introduction

This chapter focuses on the development of the testbed for frequency offset TR modulation. The theory on transmit reference modulation was given (Section 2.5), but baseband operation was described. In practice, transmission is carried out in the passband and therefore it is necessary to briefly explain the passband scheme that was utilized. A description of the design and operation of the testbed will then be given and finally, the limitations of the testbed will be presented.

4.2 Transmit Reference Modulation in the Passband

The passband scheme used in this work was based on simply upconverting the generated baseband transmit reference signal by multiplying it with a high frequency carrier. It is shown in Figure 4.1.

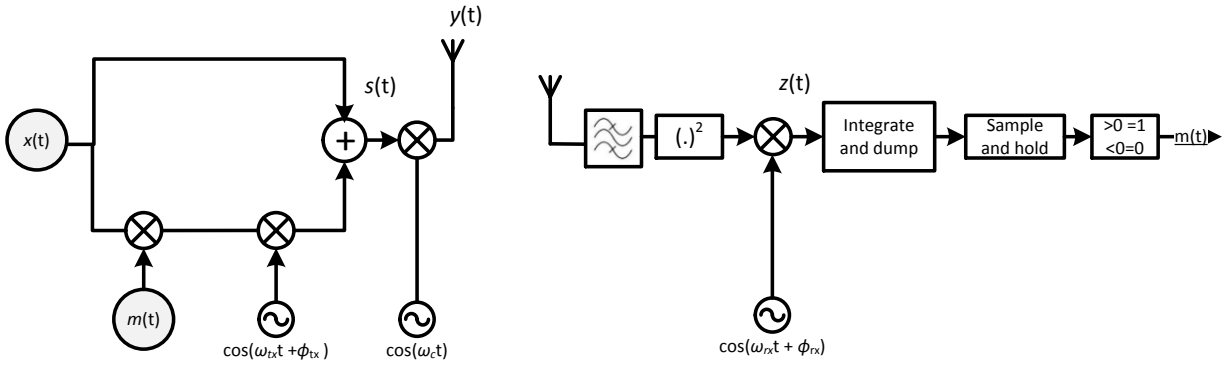


Figure 4.1 Passband scheme for transmit reference operation

The basic operation is similar to the baseband operation described in Section 2.5. The transmitted signal is given by (3.1). At the receiver, the signal is bandpass filtered to limit the channel noise to the bandwidth of the transmitted signal. The filter is centred at the carrier frequency, f_c.

After squaring and multiplying by the frequency offset, the resulting signal is given by

$$z(t) = y^2(t)\cos(\omega_{rx}t + \varphi_{rx}) \tag{4.1}$$

$$= s^2(t)\cos^2(\omega_c t)\cos(\omega_{rx}t + \varphi_{rx}) \tag{4.2}$$

$$= \frac{s^2(t)}{2}[\cos(\omega_{rx}t + \varphi_{rx}) + \cos(2\omega_c t + \omega_{rx}t + \varphi_{rx}) + \cos(2\omega_c t - \omega_{rx}t + \varphi_{rx})] \tag{4.3}$$

The first term in (4.3) is comparable to Equation (2.6), but with a factor of 0.5 arising from squaring the high frequency carrier. Hence it can be seen that despreading still occurs, but with additional signal-signal cross terms which will appear around $2\omega_c$ due to the high frequency carrier.

4.3 Realization of the Testbed

The system design for the testbed is shown in Figure 4.2.

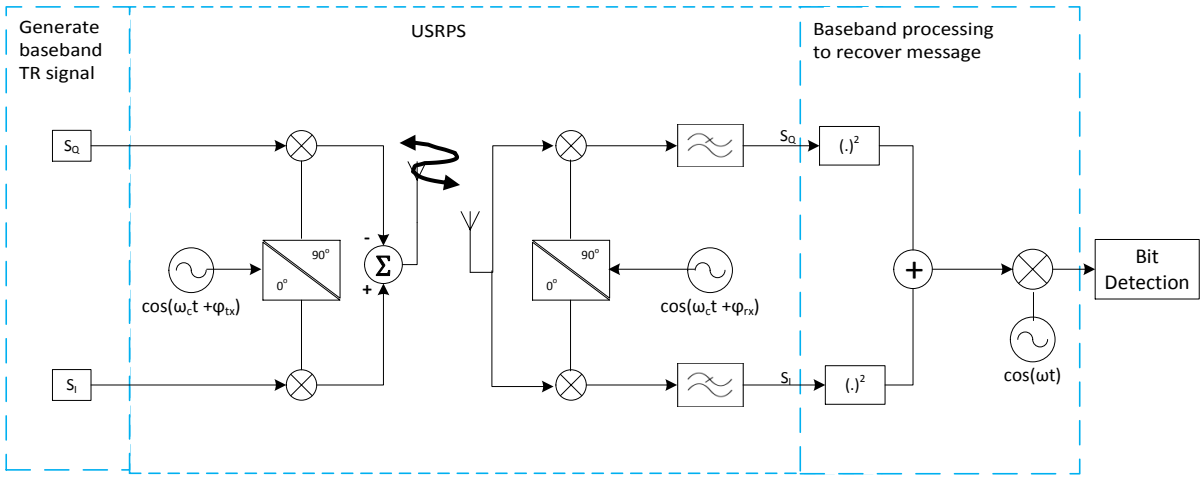


Figure 4.2 Block diagram of testbed

Two USRPs are used because the available model (2921) was not capable of simultaneous transmit and receive. Furthermore, this was a preferred option because it provided additional flexibility for example in scenarios where it would be required to carry out wireless transmission

or to have variable separation distance between the transmitter and receiver. The complex baseband TR signal is generated on the host computer— as described in Section 3.3.1— and then sent to the USRP and transmitted. After reception, the baseband processing is done as described in Section 3.3.3 to 3.3.4.

A key difference to note in the implementation with the USRPs is that despreading does not occur at RF as expected but instead has to be done at baseband. This is because when the USRP is used as a receiver, it first downconverts the signal to baseband and returns the quadrature components; hence there is no way to do the despreading at RF.

4.4 Configuration Steps for the USRP

The USRP can be configured as a transmitter or receiver through a set of controlling VIs available in LabVIEW, provided the driver software is installed.

4.4.1 The USRP in transmit mode

Four steps are required in order for the USRP to act as a transmitter. They are shown in Figure 4.3.

1. Opening a Transmit (TX) session

It accepts an IP address for a specific USRP, opens a connection to it and returns a session handle which is used to identify the connection in subsequent USRP VIs.

2. Configuring the signal

In this step, the transmit parameters are defined such as the required IQ rate, carrier frequency, transmit gain and the transmit port to use. If the specified value is out of the range supported by the device, or within range but not possible, the VI adjusts the input value and returns the adjusted value.

3. Writing data to the USRP

This VI accepts as input the complex baseband data of maximum amplitude 1, to be sent to the USRP for transmission. If any IQ values in the input data result in a magnitude greater than 1, the USRP driver clips the value to 1 and the signal is distorted. It is therefore important to ensure all values are appropriately scaled before providing them as inputs to the USRP. For continuous data transmission, the write function is done in a loop i.e. the input data is repeatedly transmitted.

4. Close Session

This closes the session handle to the device that was created in step 1.

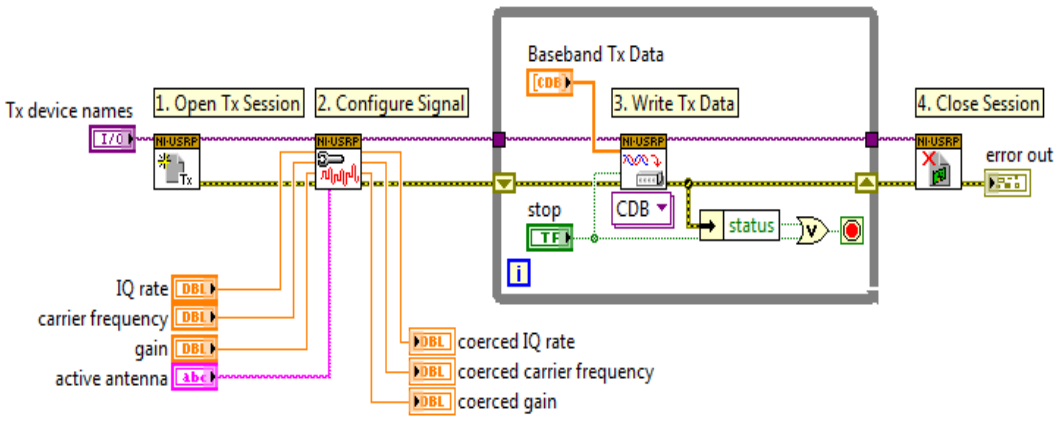


Figure 4.3: Illustration of USRP configuration steps **[18]**

4.4.2 USRP in Receive Mode

In the receive mode, the USRP requires 7 steps as has been illustrated in Figure 4.4.

1. Open RX session and
2. Configure Signal, which are similar to the transmit mode discussed in the previous section. The same parameters should be set in the transmitter and receiver for this step.
3. Initiate, which confirms that configuration is complete and that the USRP can now begin data acquisition (step 4).
4. Fetch Rx data in which the complex data is acquired.
5. Analyze and present data (in this case the baseband processing step in Figure 4.2)
6. Abort
7. Close Session, which is used to end the receive session.

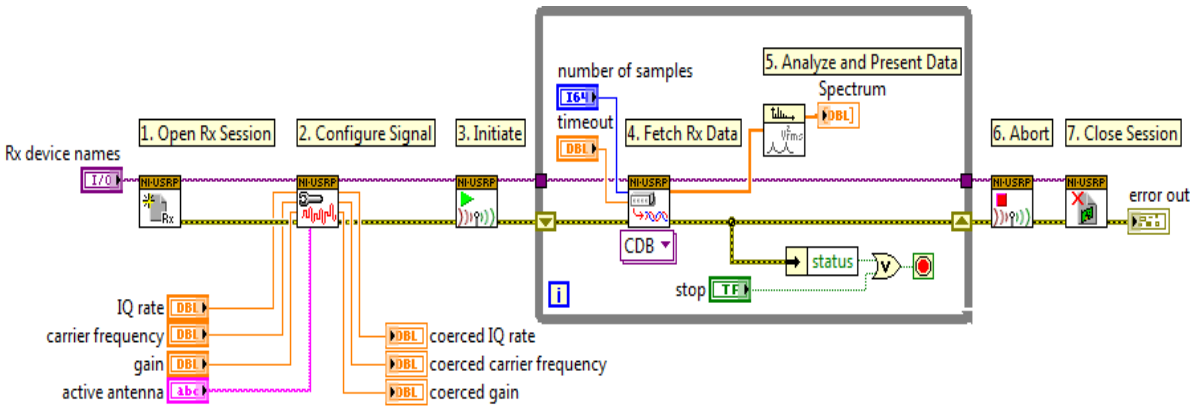


Figure 4.4: USRP configuration steps for receive mode **[18]**

4.5 Hardware Setup

When the configuration was complete, the hardware was then set up as shown in Figure 4.5.

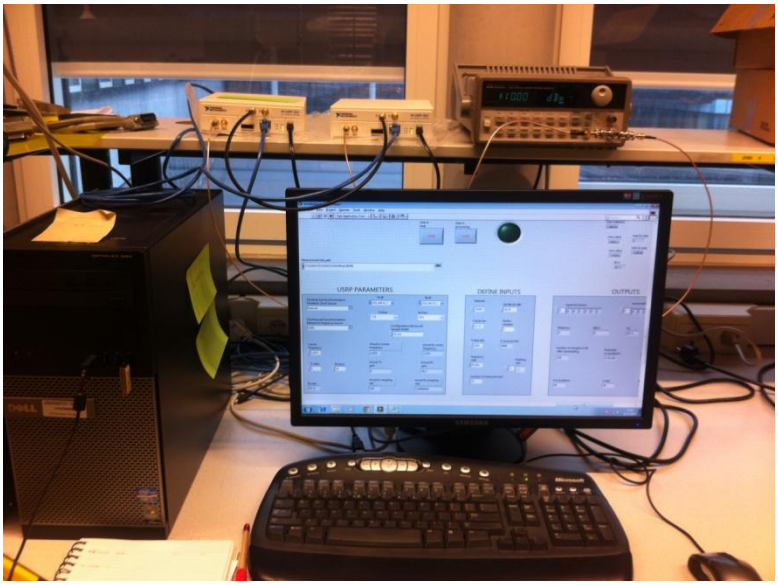


Figure 4.5 Hardware setup of the testbed

The setup consisted of a desktop computer on which LabVIEW and the USRP driver software was installed, two USRPs and waveform generator. The PC was also equipped with two ethernet cards, one for each USRP, to maximize streaming performance. The transmit port of one USRP was connected to the receive port of the second USRP using a coaxial cable. A 30 dB attenuator was also used before applying the signal to the receive port to avoid damaging the receiver. The

waveform generator was a Hewlett Packard (HP) 33120A, which was used to supply a 10 MHz reference sinusoidal signal for the USRP local oscillators. The output signal from the waveform generator was split and supplied to the REF IN port of each USRP.

4.6 Synchronization in the System

Ideally, after setting up the testbed as described, it should work correctly, assuming the receiver and transmitter are properly synchronized. However, like in any practical system this is not the case and additional steps are required. For this testbed, two issues were of importance: local oscillator synchronization and signal alignment at the receiver.

4.6.1 Local Oscillator Synchronization

It has been shown in the theory and from Figure 4.1 that for transmit reference systems synchronization of the high frequency oscillators is not necessary and only the low frequency oscillators that are used to apply the frequency offset need to be synchronized. However, it has also been highlighted that because of the implementation with the USRPs, the receiver is different from the expected passband receiver and this step is still necessary i.e. the frequency and phase of the oscillator in the receiver has to be exactly matched to the transmitter to recover the baseband quadrature components. It is possible to achieve this synchronization in the testbed in the two ways below:

a) Frequency synchronization of the RF oscillators could be done in the hardware either by applying a common reference signal to both USRPs or by configuring them in a master-slave configuration such that one USRP derives its reference signal from the other. The first method was chosen because the waveform generator was immediately available in the laboratory. The second method required the purchase of an additional cable from the equipment vendor.

b) Phase synchronization could be done in the software, basing on one of the receiver structures proposed in [8] which is described here. Recalling from (3.2), the quadrature component is zero and hence the transmitted signal is actually the in-phase component s_I, multiplied by the carrier.

At the receiver, if the two oscillators are not phase synchronized then the received quadrature components are:

$$\widetilde{s_I} = s_I \cos(\varphi_{tx} - \varphi_{rx}) \tag{4.3}$$

$$\widetilde{s_Q} = s_I \sin(\varphi_{tx} - \varphi_{rx}) \ ,$$

where φ_{tx} and φ_{rx} are the transmitter and receiver oscillator phases respectively. Squaring and adding each quadrature component gives

$$s_I{}^2 \cos^2(\varphi_{tx} - \varphi_{rx}) + s_I{}^2 \sin^2(\varphi_{tx} - \varphi_{rx}) = s_I{}^2, \tag{4.4}$$

where the trigonometric identity $\cos^2(x) + \sin^2(x) = 1$ was used.

Synchronization of the low frequency cosine used as a frequency offset could easily be done because it was generated in the software. Hence it required only wiring the output from one waveform generator to both the transmitter and receiver.

4.6.2 Signal Alignment

Correctly recovering the sent bit stream requires knowledge of the start of the signal. The signal alignment step was necessary because it was observed that the received signal was a re-ordered version of the sent data symbols (Figure 4.6c). The reason for this was not explicitly clear but it is possible that some additional timing functions are required to instruct the USRP when to start acquisition so that the write and fetch operations are exactly synchronised for example through knowledge of the propagation delay. Aligning the signal was important for two reasons:

(a) the BER calculation algorithm simply compares the array of sent and received bits, which means that the array of recovered bits had to be ordered in exactly the same way or else the algorithm would fail.

(b) it facilitated bit timing which was crucial for the bit detection. Since the number of samples per symbol was known, identifying the initial start of the first symbol was a way to easily achieve synchronization for all the symbols.

The method used in this work was based on a common method applied in packet transmission in which a synchronization sequence is included in the message. At the receiver, the sequence is known and by cross-correlating the received signal with this known sequence it is possible to align the signal.

The algorithm used in the testbed works in the following way. Let the signal to be transmitted be consist of 5 symbols as in Figure 4.6(a). Furthermore, let each symbol consist of N samples, so that the total number of samples is $5N$. When the USRP write function (Section 4.4.1) is put in a loop, continuous transmission can be achieved and the same signal is sent multiple times given by Figure 4.6(b).

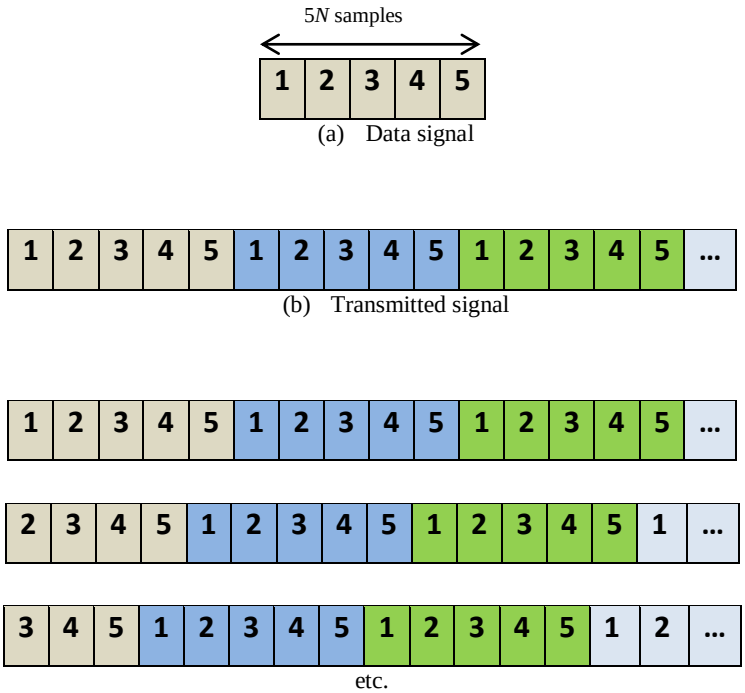

5N samples
1 2 3 4 5
(a) Data signal
1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 ...
(b) Transmitted signal
1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 ...
2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 ...
3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 ...
etc.
(c) Examples of possible received versions of the transmitted signal

Figure 4.6 Illustration of signal alignment algorithm utilized in the testbed

At the receiver, it is required to recover the transmitted symbols in order. Furthermore, the user is required to specify how many samples should be fetched by the USRP. It can be observed that

fetching ≥3 times the number of samples ensures that there is always a correctly ordered version of the message in the received signal. Since the transmitter and receiver were implemented in the same VI, the data signal is known and could be used as the synchronization sequence. Thus by correlating 4.6(a) with the corresponding received version in 4.6(c), a peak will be obtained in the correlation function when alignment is achieved. Since the number of samples in the information signal is known, it can then be extracted from the received signal.

4.7 Measurement Settings and Methodology

The measurements were carried out by transmitting 5,000 message bits, at a carrier frequency of 2.45 GHz. The resulting TR signal was repeatedly transmitted and received. Noise of varying power was added to the received complex baseband signal in the software after downconversion resulting in E_b/N_0 values from (0 – 30). For each value of E_b/N_0, a different message bit stream was sent for each measurement since the PN data sequence was generated by a random initial seed. In addition, several measurements were taken per E_b/N_0 and the average BER computed.

4.8 Challenges/Limitations of the Testbed

The developed testbed has not achieved as much flexibility as was hoped for because of some limitations which are highlighted below.

4.8.1 Achievable IQ Rates

The USRP is only capable of a finite number of IQ rates. Some input values are adjusted to values supported by the USRP which in turn meant that not all bandwidths were possible. In this work, the IQ rate is chosen as NB, where N is the number of samples in a chip period and B is the spreading bandwidth. For example if $N = 2$, IQ rates of 6 and 8 MS/s are coerced to 5.88 and 7.69 MS/s so bandwidths of 3 MHz, 4MHz could not be used. It was for this reason that it was preferred to vary the processing gain by varying the datarate rather than maintaining a constant datarate and varying the spreading bandwidth.

4.8.2 Maximum Achievable Bandwidth

The maximum bandwidth that can be achieved in the testbed is limited by the maximum USRP IQ rate and the choice of N.

From the vendor specifications, the maximum IQ rate is 25 MS/s to achieve a double-sided signal bandwidth of 20 MHz. In the setup, the maximum rate that has been achieved is 10 MS/s. Higher streaming rates were not possible because the performance of the system became intermittent, in that the signal was acquired with missing samples or no samples were fetched at all. This was attributed to computer hardware requirements by the vendor (National Instruments) and also the fact that the waveform size being transmitted was large. The table below compares the specifications of the PC used versus some recommended settings by the vendor [19] to achieve the maximum IQ rate of the USRP.

Table 4-1 Comparison of vendor recommended PC specifications versus available PC specifications

Vendor recommended specifications	**Specifications of PC used**
Dell i7 PC	Dell i7
Quad core	Quad core
16 GB RAM	8 GB
Intel network adapter recommended	- Network adapter 1: Intel - Network adapter 2: TP-link
Other settings - High performance power plan in windows - Disable power savings options - Changing registry key - Change driver options such as buffer sizes	 - Done - Done - Not done - Not done

The maximum achievable spreading bandwidth (single- sided) in the testbed is then determined by $10/N$.

4.8.3 Achievable Spreading Factors

In order to obtain higher processing gains, the datarate had to be lowered compared to the carrier bandwidth. It has been shown in Section 3.3.3 that the number of samples per symbol is

dependent on the spreading factor. Higher spreading factors meant that more samples were required which slowed down the program execution and also made the signal acquisition intermittent, because the USRP is in turn required to acquire a large number of samples per fetch.

4.8.4 Manual measurement process

The operation of the measurement VIs is relatively manual requiring the user to start the VI for each value of E_b/N_0, which is not ideal when many measurements are being carried out. The operation of the VIs will be explained in Appendix C. Due to time constraints, it was not possible to automate the measurements.

Chapter 5

Results and Discussion

This chapter presents the results that were obtained from measurements on the testbed and a discussion about them in comparison to the theoretical values.

5.1 Measurement Results and Settings

The results obtained are given in the section that follows.

5.1.1Initial Measurement Results

These were carried out with the settings presented in Section 4.7.

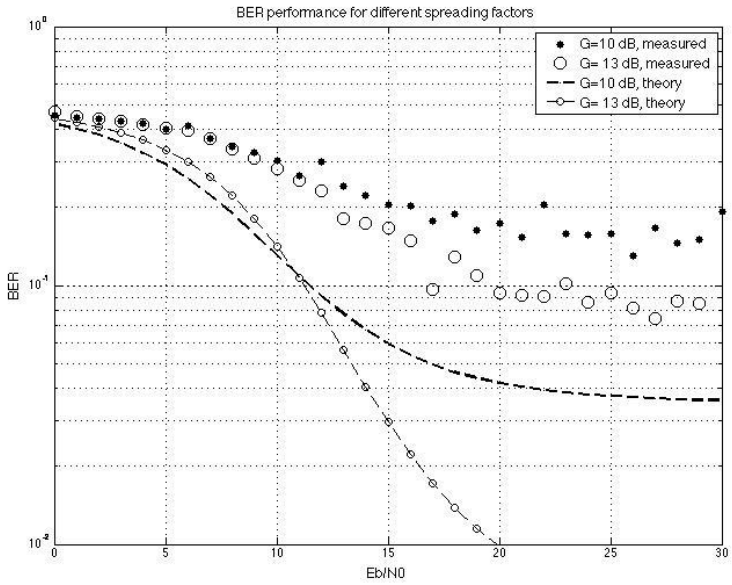


Figure 5.1 Performance for different processing gains with AWGN as carrier, 1 MHz spreading bandwidth and *N*=2

Figure 5.1 illustrates the initial measurements that were carried out. The dashed lines represent the theoretical curves while the measured values are illustrated with markers. Generally, the performance was worse than predicted by theory which was attributed to the following factors:

- it was observed that there were periodic spikes in all the BER measurements which raised the average values considerably higher. A sample measurement run, consisting of 100

measurements for one value of E_b/N_0 is shown in Figure 5.2. It can be seen that average BER is almost 10 times the actual BER.

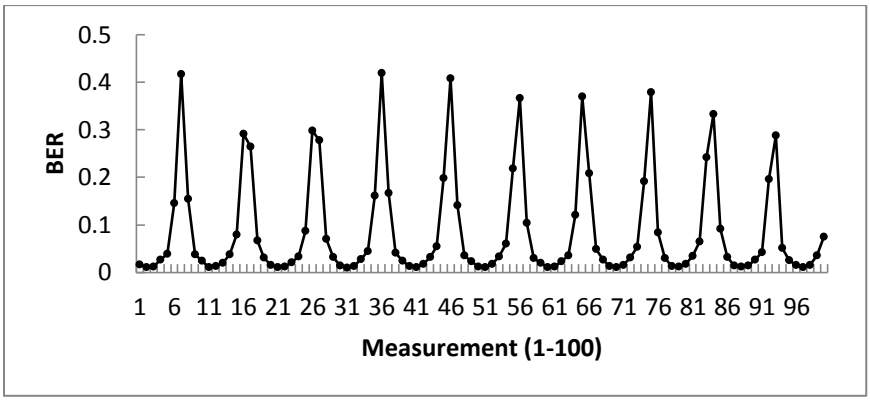


Figure 5.2 Illustration of periodic error obtained in measured values

The reason for this was found to be that the transmitter and receiver local oscillator frequencies were not synchronized. The carrier frequency is realized in the hardware as a combination of the local oscillator frequency and a digital signal processor [20]. The local oscillator is set as close as possible to the carrier frequency and then the remainder is obtained digitally. Initially, the measurement setup was as shown in Figure 4.5, but without the waveform generator. The reference signal for the local oscillators was derived from each individual USRP clock, which meant that each LO could lock at a different frequency, and the received quadrature components would be modulated by a low frequency sinusoid, whose frequency was equal to the frequency difference between the transmitter and receiver LOs.

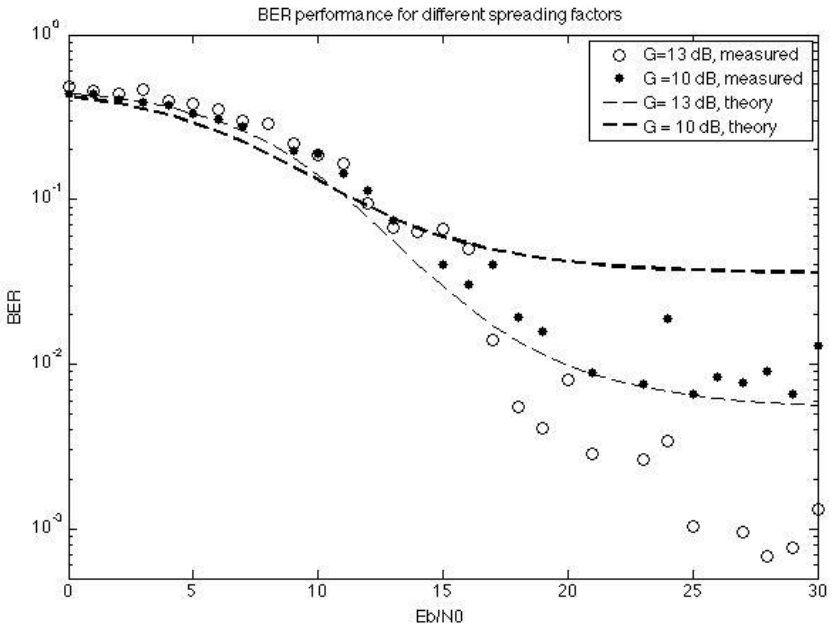
- Furthermore, *N*=2 was a poor choice because it meant that it did not account for the doubled signal bandwidth after squaring.

5.1.2 Final Measurement Settings

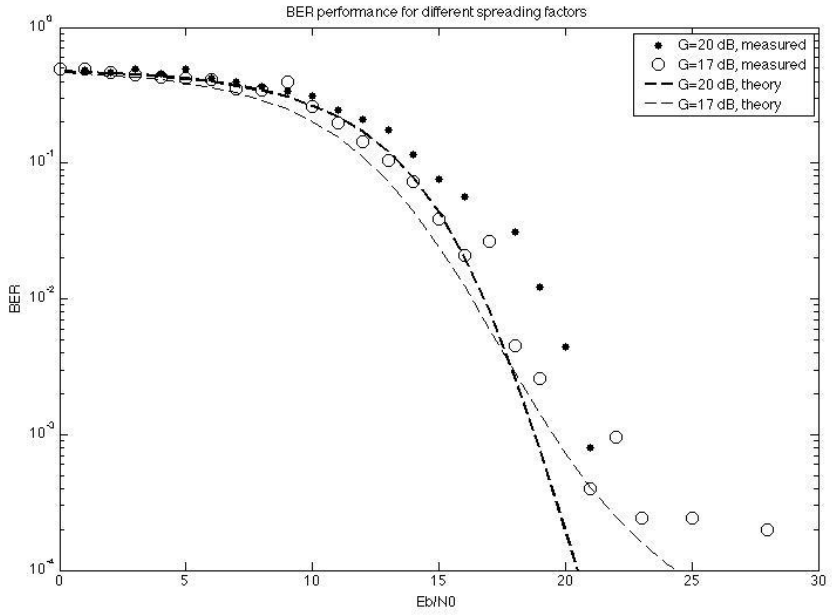
In light of the above, the measurements were repeated with the new setup including the waveform generator supplying a common reference signal and N =5. The spreading bandwidth was 1 MHz, resulting in a sampling frequency of 5 MS/s. Spreading factors from 10 – 100 were measured by varying the datarate from 10 kbps to 100 kbps. The frequency offset was chosen as equal to the datarate for each case. For the spreading factors of 10, 20 and 50, 10 measurements were carried out per value of E_b/N_0 and the average BER computed. For the spreading factor of 100, only one measurement was carried out because after one iteration of the measurement loop, the system performance would become intermittent as explained in Section 4.8.3.

Additionally, it was necessary to filter the generated noise.

5.1.3 Final Measurement Results



(a) 1 MHz bandwidth, datarates of 100 kbps and 50 kbps



(b) 1 MHz bandwidth, datarates of 20 kbps and 10 kbps

Figure 5.3 BER performance for different spreading factors in 1 MHz bandwidth

5.2 Discussion

In Figure 5.3, the theoretical curves are shown by dashed lines while the measured values are shown by markers.

Generally, the trend follows that predicted by theory in that higher processing gains exhibit a steeper fall off at higher values of E_b/N_0, but perform worse than lower spreading factors at low E_b/N_0. The measured values are shifted by about 2 dB which may be due to additional noise in the system that was not accounted for.

Error floors have also been observed for the lower spreading factors as expected from theory. However, the measured error floors are much lower than predicted by theory. The reason for this has not been understood. At the higher spreading factors, the error floor has not been obtained. From the theory, it is expected that an error floor is reached at 2.666×10^{-5} and 1×10^{-8} for the spreading factors of 17 and 20 dB respectively, which required transmission of a large number of message bits. This could not be done partly due to memory limitations and the fact

that the waveform size would become large which in turn would lead to intermittent signal acquisitions.

Chapter 6

Conclusions and Recommendations

In this thesis, a hardware testbed for noise-based frequency-offset transmit-reference systems has been realized. The system has been implemented using universal software radio peripherals and the LabVIEW development platform. The transmit reference signal is generated in baseband on a PC and the digital I/Q samples are sent to the transmitting USRP over a gigabit ethernet connection. The USRPs provide the digital to analog/analog to digital conversion and the up/downconversion between baseband and RF. The received baseband signal is processed on the host computer in LabVIEW. The following conclusions and recommendations have been made.

6.1 Conclusions

The USRP is a suitable hardware platform for implementing a transmit reference system. However, a key difference arises which is that despreading does not occur at RF but rather in the baseband. This is because when using the USRP as a receiver, down conversion to baseband is done first before the signal is available for processing. It is only after this step that TR demodulation can take place. This means that synchronization between the two high frequency oscillators is still required to correctly demodulate the signal, unlike in a conventional TR system working in the passband.

Simulations and performance measurements on the hardware testbed in the presence of AWGN at the receiver have been done. In the simulation, this has been done for two different carrier signals. In the first case, a spreading PN sequence with a rectangular pulse shape has been used and the results have shown the performance to disagree with the theory. In fact, a better performance is obtained. This is attributed to the fact that the spectrum of such a sequence using rectangular pulses is not flat, but rather sinc-shaped.

In the second case, AWGN has been generated and used as the carrier. The results have shown an agreement with the theory in general – for low values of E_b/N_0, lower spreading factors have a better performance and as the E_b/N_0 increases, higher spreading factors exhibit steeper fall off. An error floor has also been obtained for the lower processing gains. The simulation

results have been shown to match well with the theory although an exact curve fit has not been obtained, which has been attributed to differences in the numerical approximations/calculations in the software versus the analytical approximations in theory.

For the practical results, the curves obtained match well with theory, but are shifted by about 2 dB which can be attributed to additional noise in the system not accounted for. Furthermore, the error floors obtained at low spreading factors have been shown to be lower than the theoretical values. The reason for this has not been explained.

Finally, the system has only been tested up to IQ rates of 10 MS/s because of PC hardware limitations which could not allow for continuous acquisition of data by the USRPs at high streaming rates. Furthermore, only few processing gains have been measured. Higher processing gains required more samples to be fetched which in some cases caused intermittent acquisitions and in other cases required longer times to measure. Also, not all streaming rates were possible by the USRP- which limits the achievable processing gains and spreading bandwidths even further.

6.2 Recommendations and Future work

The computer used in the testbed met most of the requirements specified by the vendor. However, more adjustments should be made to make the testbed more flexible such that higher bandwidths can be achieved for example increasing the RAM to 6 GB, installing a second intel ethernet card and adjusting the recommended settings that were recommended by the vendor. When this has been done, more measurements should be carried out for different processing gains, using larger carrier bandwidths if required and transmitting a larger number of bits.

More values of the spreading factor should also be measured and compared with theory to fully explain why the measured error floors are lower than in theory.

Pulse shaping should be applied to the message signal. In this work, a rectangular pulse has been used which is not ideal because it is well known that the resulting signal possesses a large bandwidth.

The measurements have been done using AWGN generated in the software. Real channels are not AWGN channels and hence different kinds of noise should be investigated, as well as measurements when external noise is introduced. Also only a wired setup has been used.

Measurements should be done in a wireless channel since the WALNUT project, under which this Master assignment was taken, is focusing on a wireless system.

Finally, the VIs should be optimized for better performance in the following ways:

- A lot of processing is done on the signal, which makes the processing loop run much slower than the transmit and data acquisition processes, especially at higher spreading factors. One of the causes of this is the way in which the signal alignment/bit timing has been implemented as explained in Section 4.6.2, which requires fetching three times more samples than would otherwise be required. Some research into a better way to implement it such as making use of the synchronization VIs and the clock reference port available with the USRP should be undertaken.
- The measurement VIs should also be improved to make the measurement process more automated.
- The maximum achievable bandwidth has been limited in part by the required sampling rates. However, a large sampling rate is only required before the TR demodulation. Since the baseband processing has been implemented in a separate loop than the signal acquisition as shown in Appendix B, it can be possible to sample the signal at lower sampling rates in the transmitter and signal acquisition stages, and then only upsample it (using a resampling block in LabVIEW) just before squaring. This would greatly improve the processing time and the achievable IQ rates and spreading bandwidths.

References

[1] Chris Townsend, Steven Arms, and MicroStrain Inc., "Wireless Sensor Networks: Principles and Applications," in *Sensor Technology Handbook*, Jon S. Wilson, Ed.: Elsevier, 2005, ch. 22.

[2] Mosa Ali Abu-Rgheff, *Introduction to CDMA Wireless Communications*, 1st ed. Great Britain: Academic Press, 2007.

[3] Marvin K. Simon, Jim K. Omura, Robert A. Scholtz, and Barry K. Levitt, *Spread spectrum Communications Handbook*, 2nd ed. United States of America: McGraw Hill Inc., 2002.

[4] Jaap C. Haaartsen, Arjan Meijerink, Abdel Bekkaoui, Alper Taban, and Joe L. Tauritz, "Novel wireless modulation technique based on noise," in *11th Annual symposium on Communications and Vehicular Technology in the Benelux*, Ghent, 2004.

[5] University of Twente, Telecommunication Engineering group. Website of the TelecommunicationsEngineering Group at the University of Twente. [Online]. http://www.utwente.nl/ewi/te/projects/SRR/walnut/

[6] A Meijerink, G. H. L. M Heideman, and W. van Etten, "Balanced optical phase diversity receivers for coherence multiplexing," *IEEE/OSA Journal of Lightwave Technology*, vol. 22, no. 11, November 2004.

[7] A. Bekkaoui, "Link Performance of the time offset transmitted reference system," Master Thesis, University of Twente, Enschede 2004.

[8] X. Shang, "New radio multiple access technique based on Frequency Offset Division Multiple Access," Master Thesis, University of Twente, Enschede 2004.

[9] A. Taban, "Microwave radio testbed for a new modulation scheme based on time offset," Master Thesis, University of Twente, Enschede 2004.

[10] Wietse Balkema, "Realization and characterization of a 2.4 GHz radio system based on Frequency Offset Division Multiple Access," Master Thesis, University of Twente, Enschede 2004.

[11] A. Meijerink, M. J. Bentum, and W. G. Scanlon, "Noise-based frequency offset modulation in wideband frequency-selective fading channels," in *16th Annual Symposium on communications and Vehicular Technology in the Benelux*, Louvaine-la-Neuve, 2009.

[12] I. Bilal, A. Meijerink, and M. J. Bentum, "Analysis of in-band interference in noise-based frequency-offset modulation," in *IEEE 25th Annual Symp. on PIMRC*, Washington D.C., USA, 2014, (To be published).

[13] Simon Haykin and Simon Moher, *Introduction to Analog and Digital communications*, 2nd ed. United States of America: John Wiley & Sons, Inc., 2007.

[14] National Instruments. Website of National Instruments. [Online]. http://sine.ni.com/ds/app/doc/p/id/ds-355/lang/nl

[15] National Instruments. Website of National Instruments. [Online]. http://www.ni.com/pdf/manuals/375717e.pdf

[16] J. C. Haartsen, X. Shang, J. W. Balkema, and A. Meijerink, "A new wireless modulation scheme based on frequency offset," in *12th annual Symposium on Communications and Vehicular Technology in the Benelux*, Enschede, 2005.

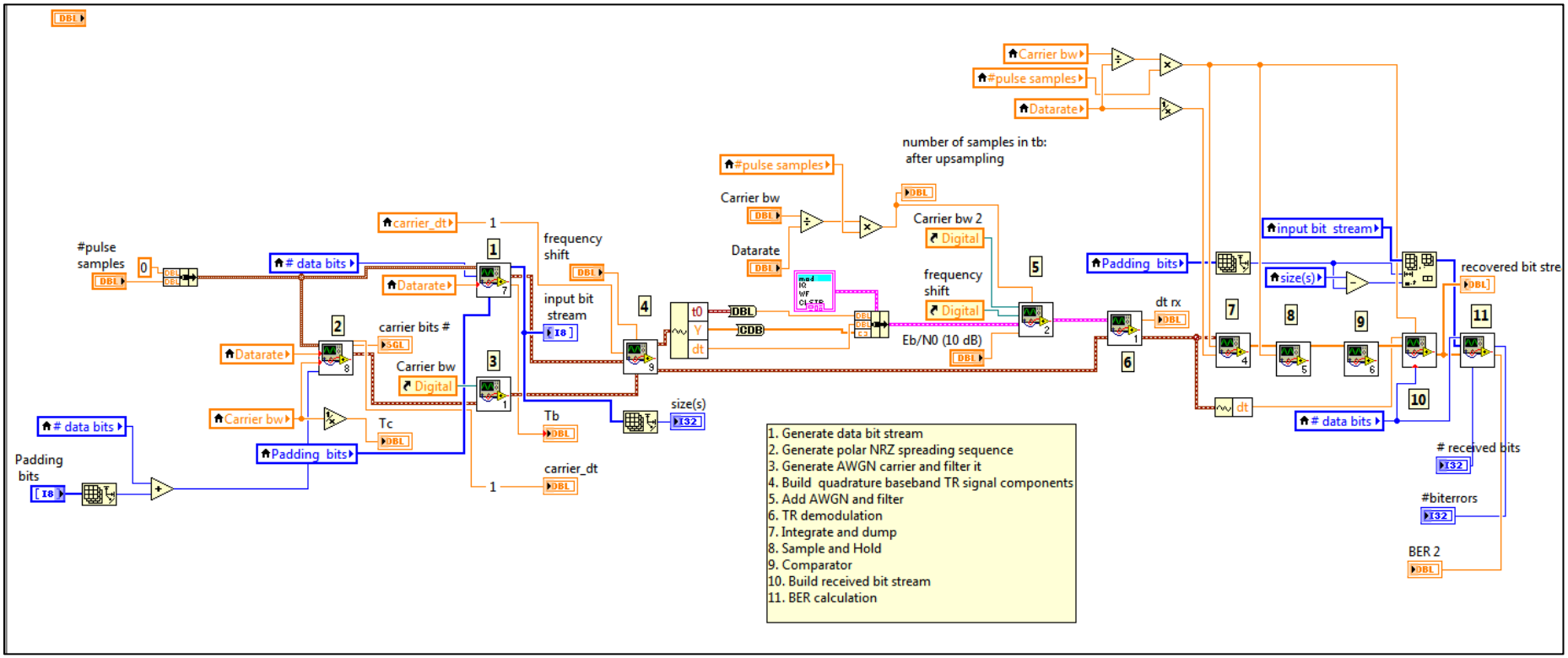
[17] National Instruments, LabVIEW context help file for AWGN Block, 2012.

[18] National Instruments, NI-USRP Help file.

[19] National Instruments. (2012, December) [Online]. http://forums.ni.com/t5/USRP-Software-Radio/Increase-the-IQ-Rate-of-USRP-NI-USRP-1-1-Driver/td-p/2256546

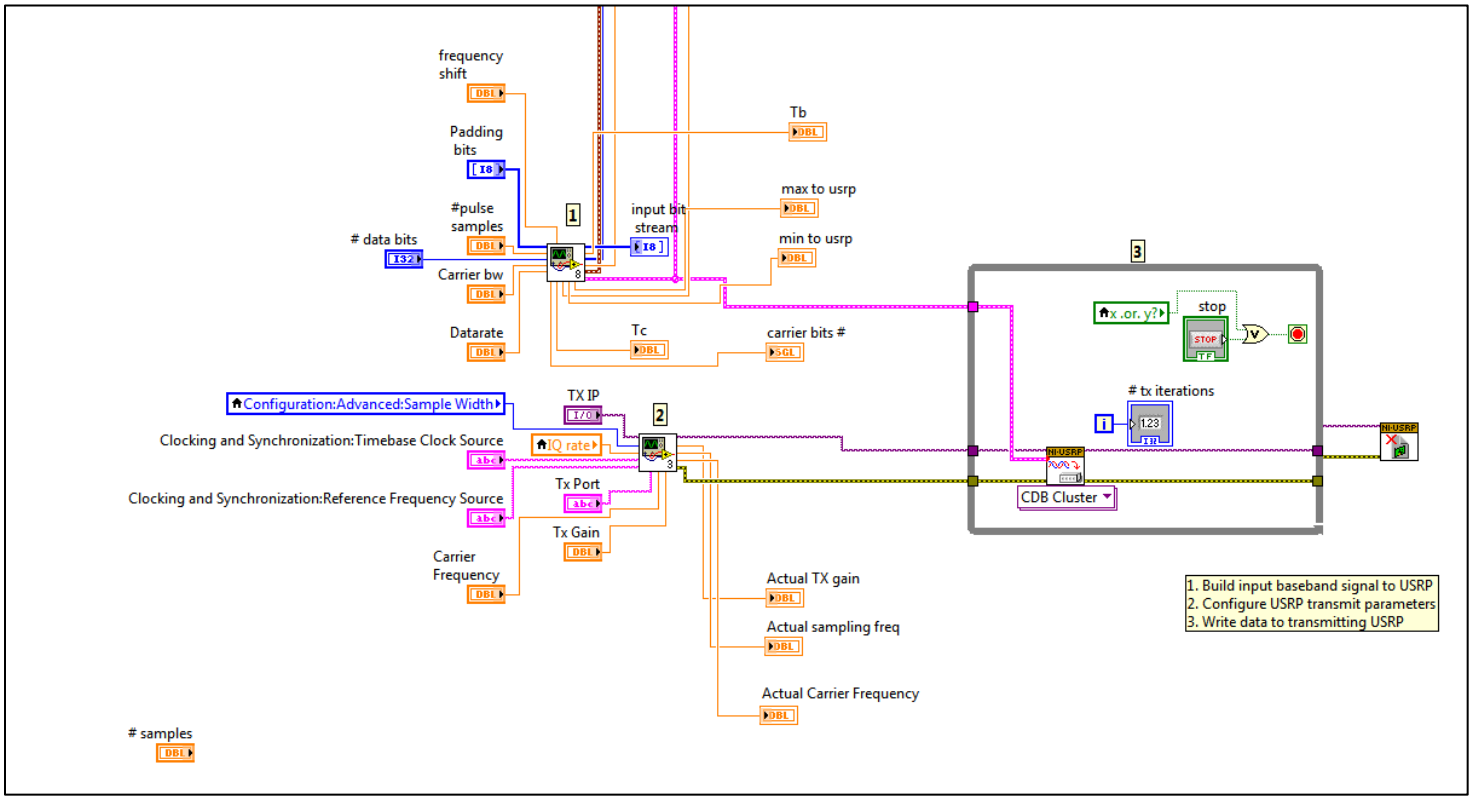
[20] National Instruments. (2012, April) N.I USRP 1.1 help manual. [Online]. http://zone.ni.com/reference/en-XX/help/373380B-01/usrppropref/pniusrp_lofrequency/

Appendix A Block Diagram of Simulation



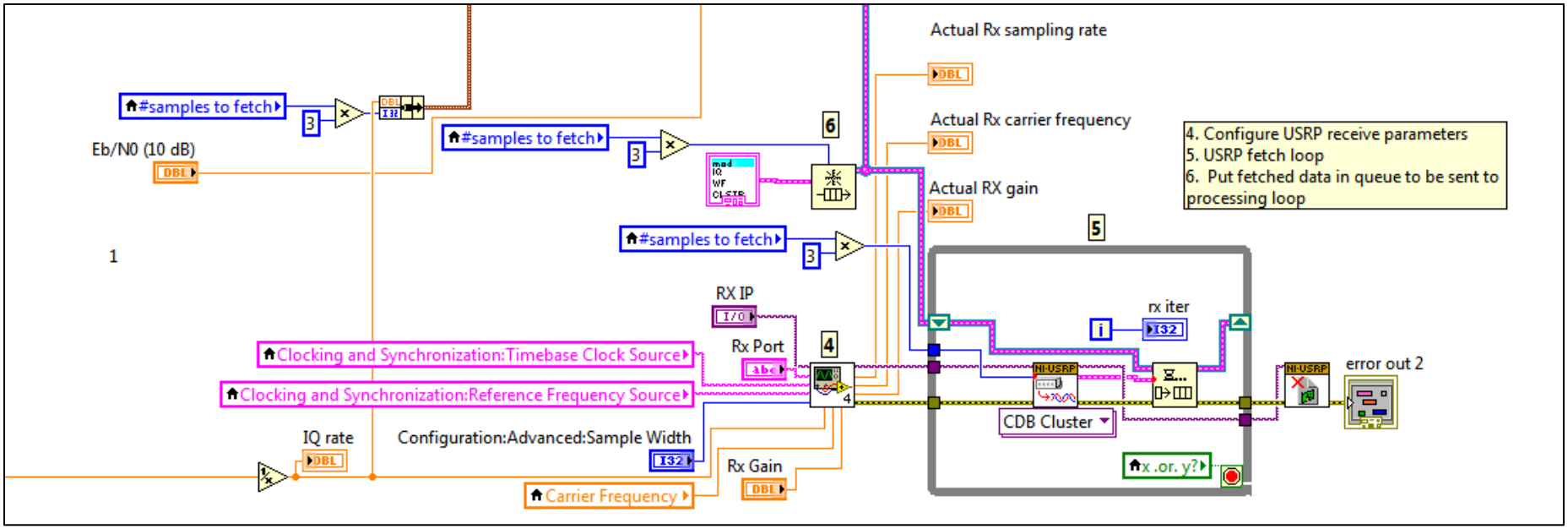
Appendix B Schematic Block Diagram of Testbed

B.1 Transmitter

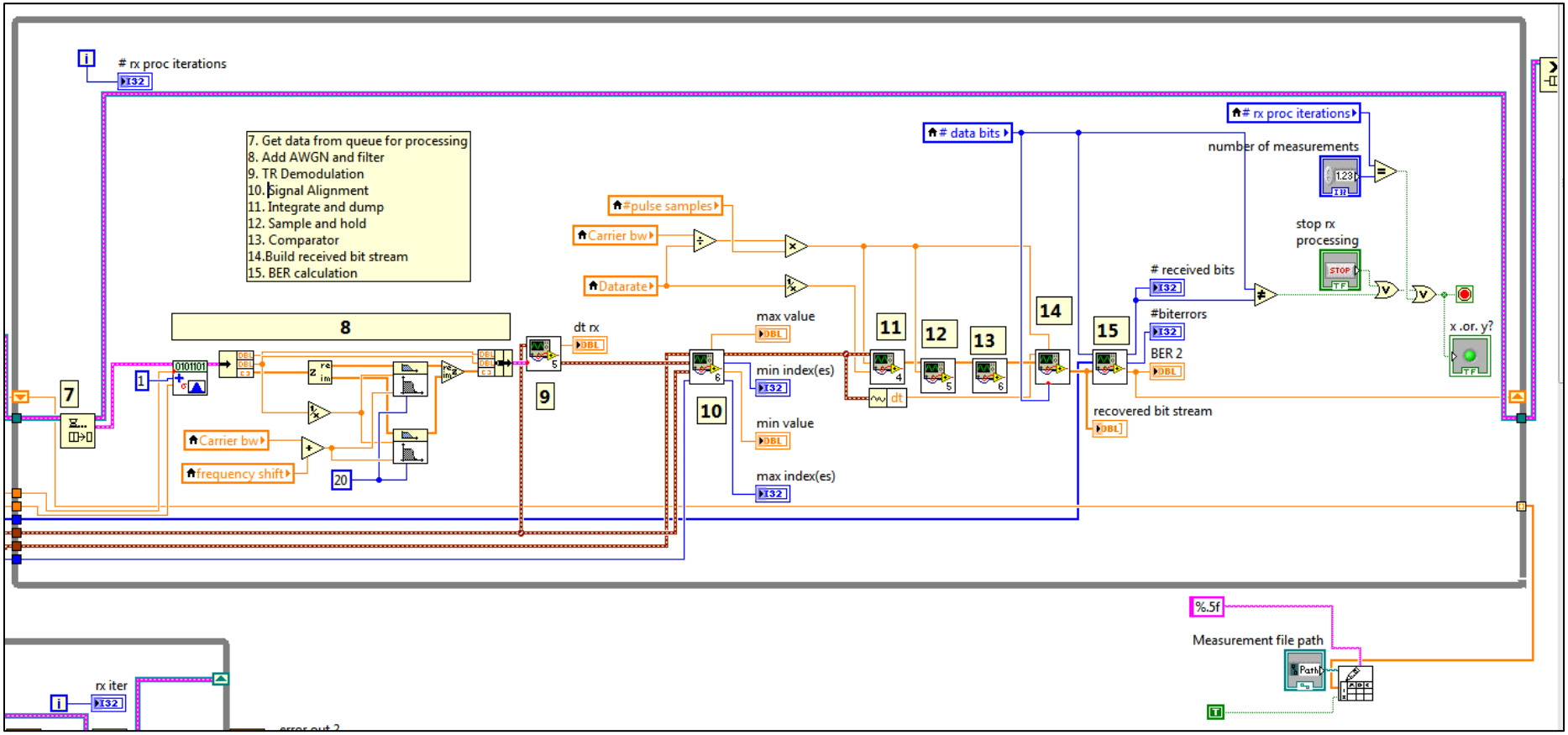


B.2 Receiver

B.2.1 Receiver Signal Acquisition



B.2.2 Receiver Signal Processing



Appendix C User Manual

This Appendix describes the way the developed VIs should be utilized.

Pre-requisites and equipment:

- A computer running windows operating system, equipped with two ethernet cards, with at least 8 GB RAM and on which LabVIEW version 2010 or later, NI modulation toolkit 4.3.3 or later and the NI USRP driver is installed.
- 2 NI USRP 2921s and accessories.
- A standard waveform generator waveform generator.
- Basic LabVIEW knowledge is required.
- Basic USRP configuration knowledge is required. The user is referred to the USRP user manual for this step.

C.1 Simulation VIs

These VIs were developed as a "proof of concept" step to verify that the system would work correctly in theory before implementing the hardware testbed.

C.1.1 Final_simulation_cplxbaseband.vi

This is the main VI to run and is dependent on the following sub-VIs

I. **Create_datastream.vi** and **Generate_20_MHz_carrier.vi**

These VIs generate a PN bit sequence of 0s and 1s, maps it to a polar sequence of ±1, and builds the data waveform using rectangular pulses.

II. **Build_baseband_components_tr_signal.vi**

In this VI, a cosine waveform is generated and together with the message, upsampled to the carrier sampling rate and the baseband TR signal is built up as in equation (3.2).

III. **Integrate_and_dump.vi**

This VI accepts an input waveform and performs a discrete integration while resetting the integrator output to zero at every multiple of the bit period.

IV. **Sample_and_hold.vi**

Samples the integrate-and-dump-output at every multiple of the bit period and maintains the value until the next sampling instant.

V. **Comparator.vi**

Accepts as input the sample and hold output and returns +1 if the value is greater than zero and -1 otherwise.

VI. **Received_bits.vi**

In this VI, the recovered bit stream is generated. The padding bits are deleted, and ±1 values are mapped back to 1 or 0 respectively.

VII. **BER calc.vi**

Compares the received bits and sent bits and returns the number of errors and bit error rate.

The user is required to define the following inputs:

- **Datarate** is the message bit rate.
- **Carrier bw** is the bandwidth of the spreading signal.
- **#pulse samples** is the number of samples in the rectangular pulse.
- **# data bits** is the number of data bits to generate.
- **Frequency shift** is the required frequency offset.
- **Padding bits** are the extra bits to be appended to the message to account for missing samples as explained in Section 3.3.
- **Required Eb/No** is required for the addition of AWGN to the signal. The variance of the noise to be added to the signal is calculated in such a way as to meet the defined E_b/N_0. See Section 3.3.2.

The following outputs are displayed on the front panel:

- **Input bit stream** is an array of the input bits.
- **Recovered bit stream** is an array of the recovered bits.
- **Tb** is the bit period.
- **Tc** is the chip period.
- **#received bits** is the number of received bits.
- **BER** is the calculated bit error rate.
- **#bit errors** is the number of bits received in error.

- **Carrier bits** is the number of carrier bits.
- **Carrier_dt** is the sampling period of the carrier waveform.
- **Number of samples in Tb after up sampling** is the samples per symbol after upsampling the message waveform to the carrier sampling rate.

Note:

- $\#\ \text{carrier bits} = (\text{number of data bits} + \text{number of padding bits}) \times \text{spreading factor}.$
- #received bits should be equal to the number of data bits, else the BER is returned as NaN. It is an indicator that the VI did not correctly run due to some errors.
- Number of samples in Tb after up sampling is the metric used by the integrate-and-dump, sample and hold and the comparator to identify the bit period. It is determined by the #pulse samples and spreading factor from (3.4). It should be an even number because the AWGN which is used to add noise to the received waveform requires the samples per symbol to be an even number. It is required as input to this block in order to calculate the energy per bit.
- By default and AWGN carrier is connected as the required carrier. To use the polar NRZ carrier, simply connect the output of the `Generate_carrier` subVI to the `Build_baseband_components_tr_signal` subVI.
- Waveform graphs and PSD blocks can be connected at any point if required to visualize the waveforms and spectra.

C.1.2 Final_simulation_cplxbaseband_formeasurements

This VI should be used for simulation measurements.
It performs the same as the previous VI except for the addition of a while loop which operates in the following way.

- The E_b/N_0 value should be left as the default value of 0 and the user is required to define a maximum E_b/N_0.
- For each run of the VI, a data bit stream is generated, the TR signal built and the E_b/N_0incremented in steps of 1 from 0 to the maximum E_b/N_0. The BER for each value of $E_b/N_0$is written to a spread sheet file, whose path and file name should be defined e.g. *C:\Users\S1316311\Desktop\BER1*

- The user should re-run the VI as many times as the required number of measurements. For example if 5 measurements are required, run the VI 5 times. The result from each measurement run is appended to the same measurement spreadsheet file. A new message bit stream is generated for each run.
- To read the measurement file, use the VI `read_measurements.vi`. This VI reads the contents of the measurement file and writes it to an excel spreadsheet. Create a blank excel file e.g. *BER1.xlsx* in the same path/folder as the `read_measurements` VI. Enter the excel filename and the path to the measurement file in the respective dialog boxes and run the VI. The spreadsheet can then be saved to the preferred location.

C.2 Testbed VIs

The main VI to run for the test-bed is named
`Whole_final_withnoise_atrx_with_queue.vi`

It is dependent on the same sub-VIs as the simulation VIs, and the following additional sub-VIs:

- **`Configure_usrp_transmit.vi`**

  This VI contains the `USRP_open_tx_session` and `configure_signal` sub-VIs explained in Section 4.4.1.
- **`Configure_usrp_receive.vi`**

  It contains the `USRP_open rx_session, configure_signal` and `initiate` sub-VIs explained in Section 4.4.2.
- **`Demodulate TR.vi`**

  In this VI, TR demodulation takes place as described in Section 4.6.1.
- **`Signal_alignment.vi`**

  Here, the signal alignment is implemented as described in Section 4.6.2.

The user is required to define the same inputs as the simulation VIs, as well as the following USRP inputs:

- **RX** and **TX IP** which identify the transmitting and receiving IP addresses.
- **RX** and **TX port** which identify the respective ports to which the coaxial cable or antennas are connected.

- **RX** and **TX gain**.
- **Carrier frequency**
- **Clocking and synchronization: timebase clock source**. It should be set to internal if the internal USRP clock are used or PPS in if an external reference is used. If a master-slave configuration is used, the master USRP should be set to internal and the slave USRP to MIMO.
- **Clocking and synchronization: reference frequency source**. It should be set to REF in since an external 10 MHz reference is used.
- **Advanced sample-width** should be left as the default value of 16 or choose 8-bit if required in the future to work with 8-bit samples.

The transmitter and receiver are implemented in the same VI. The quadrature components of the baseband TR signal are built up and the IQ samples written to the transmitting USRP in the USRP transmit loop. The samples are then processed as described in Section 2.3.3 and transmitted, and the signal received by the receiving USRP. The received baseband samples are fetched from the receiving USRP in the USRP fetch loop and placed in a queue. The signal processing loop then fetches the samples from the queue, adds noise and proceeds to do the TR demodulation and bit detection.

The signal processing loop runs for a specific number of iterations defined by the input parameter "*number of measurements*" and then stops when this number is reached, or the BER output is NaN or when the user terminates the loop by pressing the "*stop rx processing button*" on the front panel. In each of these three cases, the green indicator button on the front panel is turned on and triggers the termination of the transmit and rx acquisition loops.

To carry out a measurement the user should follow the steps below:

- Set up the equipment as shown in Section 4.5. A computer running LabVIEW 2010 or later, and installed with the USRP driver software and the NI Modulation Toolkit is required.
- Run the USRP configuration utility and check that both USRPs have been identified by the PC.
- Ensure that the green indicator light on the front panel is off.
- Define the required input values and a required E_b/N_0.

- Run the VI until it terminates.
- Turn off the green light; increment the value of E_b/N_0as required and repeat.
- Read the measurement values by running the `read_measurements` VI as previously described.

Note:

- Care should be taken not to swap the TX and RX IPs because the receiver has to be connected to a 30 dB attenuator to avoid damaging it.
- On the first run of the VI, sometimes an error shows up. Simply close this error and re-run it.
- If a buffer underflow/ overflow error occurs in the transmit loop when the VI stops running, simply close the error and re-run the VI.
- The user should keep watch on the indicator rx_iter and tx_iterations. Sometimes the rx loop runs faster than the tx loop. In this case, the measurement should be aborted and repeated. The user should then take note of this measurement and ignore it in the output file.
- If an error is encountered, or the signal alignment fails, the number of bits recovered will be less than the bits sent. In this case, the BER value is returned as NaN and the VI stops. Again this measurement should be noted and repeated.
- The USRP requires that the maximum amplitude of any complex waveform input to it is ±1. In these VIs, the maximum value of the input waveform is calculated and all values are divided by the maximum value. This operation is implemented in the `scale_input_to_usrp` sub-VI, which is located in the `build_input_signal_to_usrp` VI. If this is not done, the USRP driver clips all values with amplitude higher than 1 and the signal is distorted. This should be taken care of in the event that the input waveform is changed.
- To view the building blocks of a sub-VI/ make any modifications, simply double click the sub-VI icon, and on subsequent the front panel, click windows>show block diagram.

Acknowledgements

I would like to express my sincere gratitude to everyone that has helped me during the 7 months I have worked on this research.

First and foremost, I would like to thank my supervisors; Mark, Arjan and Ibrahim for all your ideas, guidance and discussions during the progress meetings and throughout the entire duration of my research. I cannot thank you enough for all the time you spent reviewing my work, explaining concepts I was not very familiar with, and for the timely feedback after presentations, progress reports and during the course of writing my thesis.

I would like to thank Robert Vogt for all the help with LabVIEW and in the Lab.

I would like to thank Alex Floor from National Instruments, as well as the technical support engineers that were instrumental in helping with support questions I had about the USRP.

I would also like to thank Lilian and Eduard at the TE group for all their help with the technical and administrative stuff ranging from access cards and working space to Lab equipment and many others.

It has been an amazing experience to be part of the TE group. I am grateful to all the members of the group, not only for the social breaks during the Coffee+, but also for your feedback after my presentations, and for all that I learnt from you during the bi-weekly group presentations.

I would like to express my sincere gratitude to the University, for the UTS scholarship that was awarded to me, without which, I would not have been able to start my studies at this university.

I would also like to thank my friends at home and in the Netherlands. Thank you for the support and social diversions. To Immie, Rufina, Maggie, Lorraine and Sharon— the Netherlands would not have been the same without you! To Olgar, Brenda, Diana, Jacob, Robert and Norman— thank you for keeping home close to my heart. To Ssegga, Alfonse and Ambrose—thanks for easing my woes with "code"! And to you, Zaq, thank you for making everything brighter!

I would like to thank my brothers Nobert, Osbert and Ivan for all the emotional support and encouragement. I could not ask for better siblings.

Finally, I would like to thank my mother for all her support, but most of all, for being my inspiration. To mommy— "You are a woman, phenomenally. A phenomenal woman, that's you!"—Maya Angelou (R.I.P).